CARBONITE

Carbonite Availability and Carbonite Migrate

PowerShell Scripting Guide

Notices

Carbonite Availability and Carbonite Migrate PowerShell Scripting Guide Version 8.2.1, Friday, September 21, 2018

If you need technical assistance, you can contact CustomerCare. All basic configurations outlined in the online documentation will be supported through CustomerCare. Assistance and support for advanced configurations may be referred to a Pre-Sales Systems Engineer or to Professional Services.

Man pages are installed and available on Carbonite Availability and Carbonite Migrate Linux servers. These documents are bound by the same Carbonite license agreement as the software installation.

Contents

Chapter 1 Carbonite Availability and Carbonite Migrate PowerShell overview

Carbonite Availability and Carbonite Migrate includes Windows PowerShell cmdlets that you can use to control most Carbonite features. This guide includes all of the Carbonite cmdlets available and several sample scripts.However, this guide does not explain how to use Windows PowerShell. You should reference your Windows PowerShell documentation and the many web sites devoted to PowerShell to learn how to use and script with Windows PowerShell.

If you are looking for advanced PowerShell documentation, geared towards developers who will be programming Carbonite to interact with other systems, you may want to reference the Carbonite SDK site at https://sdk.doubletake.com/double-take-sdk-resource-guide/. The Carbonite version 8.2 PowerShell documentation on the SDK site is generated directly from Carbonite source code and is organized by contracts and namespaces. This documentation may be more suitable for advanced developers.

Carbonite Reporting Service does not support PowerShell. You cannot configure or manage your Carbonite Reporting Service server with any Carbonite PowerShell cmdlets.

The following terms and definitions will help you understand Carbonite basics. See the *User's Guide* for your Carbonite product for complete details on how that product works.

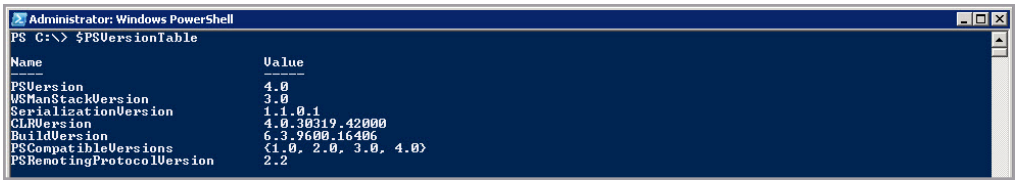
- **Source**—The source is the server that has the data you want to protect or migrate. Typically this is a machine on the production network that serves data to clients.
- **Target**—The target is the server that maintains the replicated copy of the data that is being protected on the source. Typically this is a backup server that may be local or in a remote data center. For migration jobs, this is the final destination for your data.
- **Workload**—A workload is a logical definition of the data that is being protected or migrated on the source. A workload can be a simple set of paths, for example, C:\Data or /usr. It may also be a more complex logical item that maps to multiple paths. For example, protecting a virtual machine means you are protecting multiple, specific virtual machine files, or protecting Microsoft SQL means you are protecting a SQL database and its related files.
- **Workload manager**—The workload manager is a web service that creates and configures the Carbonite workload.
- **Job**—A job is a logical unit that includes the source, target, and the workload. The job is what you create and monitor in order to protect or migrate your data.
- **Job manager**—The job manager is a web service that creates, monitors, and controls the Carbonite job.
- **Connection**—The engine connection is the underlying stream that sends the actual replicated data between the source and target servers. Jobs are higher-level objects that use the lower-level connection to protect data.
- **Architecture**—Each Carbonite installation has two services, the Management Service and Engine.
    - **Management Service**—This service is displayed as Double-Take Management Service in the Windows services list and jsvc on Linux. The service hosts the job manager and provides monitoring and control for all job types. For WCF clients, the service listens on port 6325. For non-WCF clients, the service listens on port 6326. This service offers a SOAP-based XML web services interface.

    - **Engine**—This service is displayed as Double-Take in the Windows service list and DT on Linux. The service transmits the replicated data between the source and target servers. By default this service listens on port 6320. You do not interact directly with this service.
- **Roles**—Any Carbonite installation can be a source, target, or both. The existence of a job between two servers and which direction data is being transmitted determines the server's role.
- **Security**—Carbonite enforces security by using local groups on each server where Carbonite is installed. There are two levels of security. Administrator access allows full control of Carbonite on a server, and monitor access allows read-only views of job information. When you connect to the job manager on a server, you will need to provide the credentials of a user who is a member of one of the local groups on that server.
- **Job creation**—To create a job, you will first communicate with the source to create a workload. You will then use that workload object and communicate with the target to create the job.
- **Monitoring and controlling jobs**—To monitor and control jobs, you will communicate with the job manager on the target of the job.
- **WCF client**—A Microsoft Windows client application created using Windows Communication Foundation, which is a framework for building distributed, service-oriented applications using Web services to send and receive data.

Carbonite PowerShell requirements

Carbonite requires Windows PowerShell version 4 or later.

If you are uncertain which version you have installed, check the PSVersion property of the $PSVersionTable automatic variable. This variable does not exist in PowerShell version 1, so if the variable returns nothing, you have version 1 installed. If you have version 2 or later installed, you will see a table of version information, showing your major and minor version numbers.



Installing the Carbonite PowerShell module

There are no additional steps required to install the Carbonite PowerShell module. It is automatically installed with all Carbonite installations.

Importing the Carbonite PowerShell module

You will need to import the module before you can begin using it. Use the Windows PowerShell import-module cmdlet to import the DoubleTake.PowerShell.dll module. If you completed a server or client/server installation, the module will be located in the \Service subdirectory where you installed Carbonite. If you completed a client only installation, the module will be located in the \Console subdirectory where you installed Carbonite. By default, the installation location is \Program Files\Carbonite\Replication. For example, using the default server installation lcoation, the cmdlet would be import-module "C:\Program Files\Carbonite\Replication\Service\DoubleTake.PowerShell.dll" or using the default client only installation location, the cmdlet would be import-module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll".

If nothing is returned, then the import cmdlet was successful.

The import-module cmdlet only imports a module into the current session. If you need to make the Carbonite PowerShell module available to all sessions, you will need to add an import-module cmdlet to your Windows PowerShell profile. See your PowerShell documentation for more information about profiles.

Chapter 2 Cmdlets

The following cmdlets are available in Carbonite.

Add-DtPhysicalRule

Adds a physical rule to a workload

Syntax

Add-DtPhysicalRule [-ServiceHost] <Server> [-WorkloadId] <Guid> -Path <String> [-Exclude] [-NonRecursive] [<CommonParameters>]

Add-DtPhysicalRule [-ServiceHost] <Server> [-WorkloadId] <Guid> [-Rule] <PhysicalRule> [<CommonParameters>]

Detailed Description

This cmdlet adds a physical rule to the specified workload on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your source server.	true	false
WorkloadId	Guid on page 267	Specify the workload GUID returned from the New-DtWorkload cmdlet using the workload type name parameter. See New-DtWorkload on page 110.	true	false
Exclude	Switch Parameter	Exclude the specified path from mirroring and replication. If you do not specify this option, the path will be included for mirroring and replication.	false	false
NonRecursive	Switch Parameter	Do not apply the rule to the subdirectories of the specified path. If you do not specify this option, the subdirectories of the specified path will be included/excluded.	false	false
Path	String	Specify the path on the source that contains the data that you want to protect	true	false
Rule	PhysicalRule on page 301	Use the Windows PowerShell New-Object cmdlet to create a physical rule object from DoubleTake.Common.Contract.PhysicalRule.	true	false

Outputs

ChangedItems on page 222

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName FilesAndFolders

$DtPhysicalPath = New-Object DoubleTake.Common.Contract.PhysicalRule -Property @ {Path="C:\DirName"}

Add-DtPhysicalRule -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid -Rule $DtPhysicalPath

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The script then creates a workload on the server for a files and folders job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. A new object is created from Double-Take.Common.Contract.PhysicalRule to store the physical path C:\DirName in the variable DtPhysicalPath. Finally, the physical rule is added to the workload on the server. The connections for the server object are then closed.

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName FilesAndFolders

Add-DtPhysicalRule -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid -Path "C:\DirName"

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The script then creates a workload on the server for a files and folders job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. A physical rule is then created for the path C:\DirName. The connections for the server object are then closed.

Add-DtUvraPhysicalRule

Adds a physical rule to a workload

Syntax

Add-DtUvraPhysicalRule [-ServiceHost] <Server> [-Workload] <Workload> -Path <String> [-Recurse] [-Exclude] [<CommonParameters>]

Add-DtUvraPhysicalRule [-ServiceHost] <Server> [-Workload] <Workload> [-Rule] <PhysicalRule> [<CommonParameters>]

Detailed Description

This cmdlet adds a physical rule to the specified full server to ESX appliance workload on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtUvraServer cmdlet. See New-DtUvraServer on page 108. For this cmdlet, the -ServiceHost should be your source server.	true	false
Workload	Workload on page 365	Specify the workload object returned from the Get-DtWorkload cmdlet. See Get-DtWorkload on page 82.	true	false
Path	String	Specify the path on the source that contains the data that you want to protect	true	false
Recurse	Switch Parameter	Apply the rule to the subdirectories of the specified path. If you do not specify this option, the subdirectories of the specified path will not be included/excluded.	false	false
Exclude	Switch Parameter	Exclude the specified path from mirroring and replication. If you do not specify this option, the path will be included for mirroring and replication.	false	false
Rule	PhysicalRule on page 301	Use the Windows PowerShell New-Object cmdlet to create a physical rule object from DoubleTake.Common.Contract.PhysicalRule.	true	false

Outputs

Workload on page 365

Examples

```
$DtServerObjectAlpha= New-DtUvraServer -Name alpha -UserName domain\administrator -Password
```

```
password -Port 6325

$DtApplianceObject = New-DtUvraServer -Name beta -UserName root -Password password -Port 6325

$DtApplianceHost = New-DtUvraServer -Name gamma -UserName root -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName Lvra

$DtWorkload = Get-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid

$DtRecommendedJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtApplianceObject -Source
$DtServerObjectAlpha -JobType Lvra -Workload $DtWorkload

Add-DtUvraPhysicalRule -ServiceHost $DtApplianceObject -Workload
$DtRecommendedJobOptions.JobOptions.Workload -Path "/home"

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtApplianceObject

Disconnect-DtServer -ServiceHost $DtApplianceHost
```

Three server objects are created for the source, the appliance, and the ESX server hosting the appliance, assigning the server objects to the DtServerObjectAlpha, DtApplianceObject, and DtApplianceHost variables, respectively. Then proxy and host information is retrieved for those server objects, storing the information in DtProxyInfo and DtVmHostInfo, respectively. That information is then used to retrieve the recommended job options. A rule for the path C:\Documents and Settings is added to the recommended job options. The connections for the server object are then closed.

Checkpoint-DtConnection

Creates a snapshot

Syntax

Checkpoint-DtConnection [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Checkpoint-DtConnection [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet creates a snapshot of the source replica data on the target. The snapshot of the source replica data is taken immediately on the target when the snapshot is requested.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

None

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Checkpoint-DtConnection -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable

DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Then a snapshot of the replica data on the target is taken. The connections for the server object are then closed.

Checkpoint-DtConnectionSourceQueue

Creates a coordinated snapshot

Syntax

Checkpoint-DtConnectionSourceQueue [-ServiceHost] <Server> [-CorrelationId] <Guid> [-JobIds] <IEnumerable> [<CommonParameters>]

Detailed Description

This cmdlet creates a coordinated snapshot of the source replica data on the target. This type of snapshot is not taken immediately. It is queued on the source at the time the snapshot is requested. The snapshot request operation will be transmitted in order with any other pending operations. When the snapshot operation is processed on the target, the snapshot will then be taken. This feature allows you to take a snapshot at the source time, rather than the target time. In the case of multiple servers, you can coordinate the snapshot time on the multiple source servers, rather than the varying times of the replica data on the target.

Keep in mind, since coordinated snapshots will not be taken until all of the operations ahead of the snapshot request in the source queue have been processed on the target, coordinated snapshots may be in a pending state for a while. Also the source must be accessible in order for the coordinated snapshot to be taken.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
CorrelationId	Guid	Specify a unique GUID which coordinates snapshots across jobs. This GUID is required even if you are only using one job. The GUID must be 32 hexadecimal digits and 4 dashes in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx, which is format 8-4-4-4-12. You can create this GUID on your own or use the PowerShell New-Guid cmdlet to generate one for you.	true	false
JobIds	IEnumerable	Specify an array of job GUIDs. A job GUID is returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false

Outputs

None

Examples

```
$CorId=New-Guid

$Source1Uri = "dtms://112.42.7.63:6325/"

$Source2Uri = "dtms://112.42.7.71:6325/"

$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJob1 = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object { $_.SourceHostUri -eq $Source1Uri}

$DtJob2 = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object { $_.SourceHostUri -eq $Source2Uri}

$DtJobsArray = New-Object "System.Collections.Generic.List[Guid]"

$DtJobsArray.Clear()

$DtJobsArray.Add($DtJob1.Id)

$DtJobsArray.Add($DtJob2.Id)

Checkpoint-DtConnectionSourceQueue -ServiceHost $DtServerObjectBeta -CorrelationId $CorId -JobIds
$DtJobsArray.ToArray()

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

Several variables are set including a unique correlation ID for the snapshots that will be taken as well as URI identifiers for the source servers of existing jobs. A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The jobs are retrieved from DtServerObjectBeta, and the job information is inserted into unique variables. An array of job IDs is then created. A coordinated snapshot is taken using the specified correlation ID and the jobs stored in the array. Only those jobs will have coordinated snapshots. The connections for the server object are then closed.

Close-DtWorkload

Closes the workload

Syntax

Close-DtWorkload [-ServiceHost] <Server> [-WorkloadId] <Guid> [<CommonParameters>]

Detailed Description

This cmdlet closes the workload creation process on the specified server and removes all resources associated with the workload creation process.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your source server.	true	false
WorkloadId	Guid on page 267	Specify the workload GUID returned from the New-DtWorkload cmdlet using the workload type name parameter. See New-DtWorkload on page 110.	true	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName FilesAndFolders

Close-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The script then creates a workload on the server for a files and folders job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. The workload is then closed on the server. The connections for the server object are then closed.

Confirm-DtJobOptions

Starts job validation

Syntax

Confirm-DtJobOptions [-ServiceHost] <Server> [-JobId] <Guid> [-JobOptions] <JobOptions> [<CommonParameters>]

Confirm-DtJobOptions [-ServiceHost] <Server> [-Source] <Server> [-JobType] <String> [-JobOptions] <JobOptions> [-OtherServers <Server[]>] [<CommonParameters>]

Confirm-DtJobOptions [-ServiceHost] <Server> [-CreateOptions] <CreateOptions> [<CommonParameters>]

Detailed Description

This cmdlet starts the job validation process, confirming the job options returned from the Get-DtRecommendedJobOptions cmdlet are compatible with the source and target servers you are using. View the details of the validation by using Get-DtVerificationStatus. See Get-DtRecommendedJobOptions on page 62 and Get-DtVerificationStatus on page 81. Do not confuse this process with the verification process that confirms if the data between the source and target are synchronized.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobOptions	JobOptions on page 273	Specify the JobOptions returned from the Get-DtRecommendedJobOptions cmdlet. See Get-DtRecommendedJobOptions on page 62.	true	false
Source	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103.	true	false

JobType	String	This value is the job type name.
		• **Availability for Windows jobs**
		  • **FilesAndFolders**—Files and folders
		  • **ClusterAwareFilesAndFolders**—Cluster-aware files and folders
			true	false

Name	Type	Description	Required	Pipeline Input
		• **SQL**—SQL • **ClusterAwareSql**—Cluster-aware SQL • **FullServerFailover**—Full server • **VRA**—Full server to ESX or full server to Hyper-V • **Availability for Linux jobs** • **LinuxFullServerFailover**—Full server • **Lvra**—Full server to ESX • **Migrate for Windows jobs** • **MoveDataOnlyMigration**—Files and folders migration • **MoveServerMigration**—Full server migration • **VraMove**—Full server to ESX migration or full server to Hyper-V migration • **Migrate for Linux jobs** • **LinuxMoveServerMigration**—Full server migration • **MoveLvra**—Full server to ESX migration • **Other jobs** • **Diagnostics**—Throughput Diagnostic Utility		
Other Servers	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. Specify multiple server objects in an array using the format @($server1, $server2).	false	false
Create Options	Create Options	Specify the create options available in DoubleTake.Jobs.Contract.CreateOptions. Use the Windows PowerShell New-Object cmdlet to create this object.	true	false

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}
```

```
$DtValidation = Confirm-DtJobOptions -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -
JobOptions $DtJob.Options

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The job options used by the job are confirmed, and the validation result is stored in DtValidation. The connections for the server object are then closed.

Disconnect-DtServer

Closes WCF connections

Syntax

Disconnect-DtServer [-ServiceHost] <Server> [<CommonParameters>]

Detailed description

This cmdlet closes all WCF (Windows Communication Foundation) connections that have been opened during use of the server object.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The connections for the server object are then closed.

Edit-DtJob

Edits a job

Syntax

Edit-DtJob [-ServiceHost] <Server> [-JobId] <Guid> [-JobOptions] <JobOptions> [<CommonParameters>]

Edit-DtJob [-ServiceHost] <Server> [-JobOptions] <JobOptions> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet allows you to edit an existing job that is stopped or running, using a JobOptions object that has been modified with your edited job settings.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobOptions	JobOptions on page 273	Specify the JobOptions returned from the Get-DtRecommendedJobOptions cmdlet. See Get-DtRecommendedJobOptions on page 62.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. See Get-DtJob on page 43. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet.Specify multiple job information objects in an array using the format @($JobInfo1, $JobInfo2).	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}
```

```
$DtJobForAlpha.Options.CoreMonitorOptions.TotalTimeAllowed="00:10:00"

Edit-DtJob -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -JobOptions
$DtJobForAlpha.Options

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. A job option is changed. In this case, the total time before failover is triggered is set to 10 minutes. Finally, the job options are used to edit the specified job. The connections for the server object are then closed.

Get-DtAccessLevel

Returns the security access level

Syntax

Get-DtAccessLevel [-ServiceHost] <Server> [<CommonParameters>]

Detailed Description

This cmdlet returns the Carbonite security access level for the credentials stored in the specified server object.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false

Outputs

AccessLevel on page 372

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Get-DtAccessLevel -ServiceHost $DtServerObjectAlpha
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the Carbonite security access level for that server is returned. The connections for the server object are then closed.

Get-DtActivationStatus

Returns license key validation information

Syntax

Get-DtActivationStatus [-ServiceHost] <Server> [[-Code] <String[]>] [[-AdditionalCode] <String[]>] [<CommonParameters>]

Detailed Description

This cmdlet returns the Carbonite license key validation information for the specified server. If you do not provide the code parameter, the license key currently in use will be returned. Specifying the key will return what the activation status would be if the key was applied using Set-DtActivationCode on page 136.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
Code	String	Specify the 24-character, alpha-numeric license key which applies the appropriate Carbonite license to your Carbonite server. Specify multiple keys in an array using the format @(code1, code2).	false	false
Additional Code	String	Specify any additional keys. Specify multiple keys in an array using the format @(code1, code2).	false	false

Outputs

ActivationStatus on page 211

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Get-DtActivationStatus -ServiceHost $DtServerObjectAlpha
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the validation information for the Carbonite license key assigned to the server is returned. The connections for the server object are then closed.

Get-DtAllFailoverReports

Returns all failover reports

Syntax

Get-DtAllFailoverReports [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Detailed Description

This cmdlet returns all of the failover reports for the specified job, sorted from newest to oldest.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false

Outputs

FailoverReport on page 258

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password
$DtJobInfo = Get-DtJob -ServiceHost $DtServerObjectBeta
Get-DtAllFailoverReports -ServiceHost $DtServerObjectBeta -JobId $DtJobInfo.Id
Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. All job information for all of the jobs on the server beta are stored in the variable DtJobInfo. This type of usage is common when the jobs were created in the past or if you did not store or do not know a job's ID. All failover reports for the job $DtJobInfo.Id are returned. The connections for the server object are then closed.

Get-DtBandwidthLimit

Returns bandwidth limiting configuration

Syntax

Get-DtBandwidthLimit [-ServiceHost] <Server> [-JobId] <Guid> [-ConnectionId <Guid>] [<CommonParameters>]

Get-DtBandwidthLimit [-ServiceHost] <Server> -JobInfo <JobInfo> [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet returns the bandwidth limiting configuration for the specified job .

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

BandwidthLimit on page 217

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}
```

```
Get-DtBandwidthLimit -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The bandwidth limiting configuration is then returned. The connections for the server object are then closed.

Get-DtConnectionIds

Returns connection ID

Syntax

Get-DtConnectionIds [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Detailed Description

This cmdlet returns the connection ID associated with the specified job .

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false

Outputs

ConnectionId on page 227

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtConIdForAlpha = Get-DtConnectionIds -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The connection ID for the job is then stored in DtConIdForAlpha. The connections for the server object are then closed.

Get-DtDiagnostics

Collects support diagnostics

Syntax

Get-DtDiagnostics [-ServiceHost] <Server> [-OutputDirectory] <String> [<CommonParameters>]

Detailed Description

This cmdlet collects configuration data for use when reporting problems to technical support. Because the diagnostics are gathering several pieces of information, potentially across the network to the machine where you are running the cmdlet, it may take several minutes to complete the information gathering and sending the resulting zip file to the cmdlet machine.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
Output Directory	String	Specify the full path on the machine where you are running the Get-DtDiagnostics cmdlet, to store the resulting zip file containing the diagnostics information. The specified path must already exist.	true	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Get-DtDiagnostics -ServiceHost $DtServerObjectAlpha "C:\Diagnostics"
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then diagnostics are collected for the server and stored in C:\Diagnostics. The connections for the server object are then closed.

Get-DtDnsOptions

Returns DNS options

Syntax

Get-DtDnsOptions [-ServiceHost] <Server> [-Source] <Server> [-JobType] <String> [-Workload] <Workload> [-DnsCredentials] <PSCredential> [<CommonParameters>]

Detailed Description

This cmdlet returns DNS options available between the specified source and target servers.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
Source	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103.	true	false

JobType	String	This value is the job type name.
		• **Availability for Windows jobs**
		  • **FilesAndFolders**—Files and folders
		  • **ClusterAwareFilesAndFolders**—Cluster-aware files and folders
		  • **SQL**—SQL
		  • **ClusterAwareSql**—Cluster-aware SQL
		  • **FullServerFailover**—Full server
		  • **VRA**—Full server to ESX or full server to Hyper-V
		• **Availability for Linux jobs**
		  • **LinuxFullServerFailover**—Full server
		  • **Lvra**—Full server to ESX
		• **Migrate for Windows jobs**
		  • **MoveDataOnlyMigration**—Files and folders migration
		  • **MoveServerMigration**—Full server migration
		  • **VraMove**—Full server to ESX migration or full server to
			true	false

Name	Type	Description	Required	Pipeline Input

		Hyper-V migration
		• **Migrate for Linux jobs**
		  • **LinuxMoveServerMigration**—Full server migration
		  • **MoveLvra**—Full server to ESX migration
		• **Other jobs**
		  • **Diagnostics**—Throughput Diagnostic Utility		

Workload	Workload on page 365	Specify the workload object returned from the Get-DtWorkload cmdlet. See Get-DtWorkload on page 82.	true	false
DnsCredentials	PSCredential on page 308	Specify the credential object returned from the Windows PowerShell Get-Credential cmdlet. This password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.	true	false

Outputs

DnsOptions on page 247

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha-WorkloadTypeName
FullServerFailover

$DtWorkload = Get-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid

$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtCredentialEncrypted = Get-Credential

Get-DtDnsOptions -ServiceHost $DtServerObjectBeta -Source $DtServerObjectAlpha -JobType
FullServerFailover -Workload $DtWorkload -DnsCredentials $DtCredentialEncrypted

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The script then creates a workload on the server for a full sever job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. The workload definition for the workload type and the server is then stored in the DtWorkload variable. A server

object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. User credentials are stored in a variable called $DtCredential. The script will prompt you to supply the username and password and the credentials will be encrypted. The DNS options for the two servers are then retrieved. The connections for the server object are then closed.

Get-DtEmailNotificationOptions

Returns e-mail notification settings

Syntax

Get-DtEmailNotificationOptions [-ServiceHost] <Server> [<CommonParameters>]

Detailed Description

This cmdlet returns the current e-mail notification settings for the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false

Outputs

EmailNotificationOptions on page 249

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Get-DtEmailNotificationOptions -ServiceHost $DtServerObjectAlpha
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the current e-mail notification settings for that server are returned. The connections for the server object are then closed.

Get-DtEventLogEntry

Returns a list of event log entries.

Syntax

Get-DtEventLogEntry [-ServiceHost] <Server> [-LastIndex <Int32>] [-ChunkSize <Int32>] [<CommonParameters>]

Detailed Description

This cmdlet returns a list of Carbonite event log entries for the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
LastIndex	Int32	Specify an index entry. The next index entry after the number you specify will be the starting point for the log entries returned. For example, if you specify 144 then the first log entry retrieved will be index 145.	false	false
ChunkSize	Int32	Specify the number of entries that will be returned at one time. The default number of entries is 1024.	false	false

Outputs

EventLogEntry on page 253

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Get-DtEventLogEntry -ServiceHost $DtServerObjectAlpha -ChunkSize 25
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the Carbonite event entries are displayed, in groups of 25. The connections for the server object are then closed.

Get-DtJob

Returns job information and status for the specified job on the specified server

Syntax

Get-DtJob [-ServiceHost] <Server> [[-JobId] <Guid>] [<CommonParameters>]

Detailed Description

Returns job information and status for the specified job on the specified server. To change the options of an existing job, use Edit-DtJob on page 29.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet. See New-DtJob on page 100. Specify multiple GUID objects in an array using the format @($JobGuid1, $JobGuid2).	false	false

Outputs

JobInfo [ ] on page 270

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName FilesAndFolders

$DtWorkload = Get-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid

$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtServerObjectBeta -Source $DtServerObjectAlpha -JobType FilesAndFolders -Workload $DtWorkload

$DtFnFJobGuid = New-DtJob -ServiceHost $DtServerObjectBeta -Source $DtServerObjectAlpha -JobType FilesAndFolders -JobOptions $DtJobOptions.JobOptions

$DtJobInfo = Get-DtJob -ServiceHost $DtServerObjectBeta -JobId $DtFnFJobGuid

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The script then creates a workload on the server for a files and folders job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. The workload definition for the workload type and the server is then stored in the DtWorkload variable. The recommended job options for the servers and the workload type are then stored in the variable DtJobOptions. A new files and folders job is created using the servers and the job options. The job ID is stored in the variable DtFnFJobGuid. Finally, the job information for the job is stored in the variable DtJobInfo. The connections for the server object are then closed.

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobInfo = Get-DtJob -ServiceHost $DtServerObjectBeta

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. All job information for all of the jobs on the server beta are stored in the variable DtJobInfo. This type of usage is common when the jobs were created in the past or if you did not store or do not know a job's ID. The connections for the server object are then closed.

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. This usage is common for servers that have more than one job, but you only want job information for one specific job. The connections for the server object are then closed.

Get-DtJobActionStatus

Returns the status of a job action

Syntax

Get-DtJobActionStatus [-ServiceHost] <Server> [[-JobId] <Guid>] [<CommonParameters>]

Get-DtJobActionStatus [-ServiceHost] <Server> [-Action] <ActivityToken> [<CommonParameters>]

Detailed Description

This cmdlet returns the status of a job action that has been queued for job. The first syntax returns the status of all of the actions queued for the specified job. The second syntax returns the status for the job action object specified.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	false	false
Action	ActivityToken on page 213	Specify a Carbonite job action object.	true	false

Outputs

ActivityStatusEntry on page 212

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Get-DtJobActionStatus -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. A server object is created for the server beta using the domain\administrator and

password credentials. It assigns the server object to the variable called DtServerObjectBeta. Finally, the status of the job action is returned. The connections for the server object are then closed.

Get-DtLatestFailoverReport

Returns all failover reports

Syntax

Get-DtLatestFailoverReport [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Detailed Description

This cmdlet returns the latest failover report for the specified job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false

Outputs

FailoverReport on page 258

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password
$DtJobInfo = Get-DtJob -ServiceHost $DtServerObjectBeta
Get-DtLatestFailoverReport -ServiceHost $DtServerObjectBeta -JobId $DtJobInfo.Id
Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. All job information for all of the jobs on the server beta are stored in the variable DtJobInfo. This type of usage is common when the jobs were created in the past or if you did not store or do not know a job's ID. The latest failover report for the job $DtJobInfo.Id is returned. The connections for the server object are then closed.

Get-DtLogicalItem

Returns workload logical items

Syntax

Get-DtLogicalItem [-ServiceHost] <Server> [-WorkloadId] <Guid> [-RefItem <LogicalItem>] [<CommonParameters>]

Detailed Description

This cmdlet returns the logical items associated with the specified workload.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your source server.	true	false
WorkloadId	Guid on page 267	Specify the workload GUID returned from the New-DtWorkload cmdlet using the workload type name parameter. See New-DtWorkload on page 110.	true	false
RefItem	LogicalItems on page 279	Specify an object returned from a previous Get-DtLogicalItem call.	false	false

Outputs

LogicalItems [ ] on page 279

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha-WorkloadTypeName FullServerFailover

$DtLogicalItems = Get-DtLogicalItem -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGUID

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The script then creates a workload on the server for a full sever job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. The logical items associated with the workload type and the server are then stored in the variable DtLogicalItems. The connections for the server object are then closed.

Get-DtLogMessage

Returns log messages

Syntax

Get-DtLogMessage [-ServiceHost] <Server> [-Source <String>] [-LastSequenceNumber <Int32>] [-LastTimeStamp <DateTimeOffset>] [-ChunkSize <Int32>] [<CommonParameters>]

Detailed Description

This cmdlet returns messages from the Double-Take service log file and the Double-Take Management Service log file.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
Source	String	Specify source of the log message. If no source is specified, both Double-Take Management Service messages and Double-Take service messages will be returned. To have only Double-Take Management Service messages displayed, specify MS for the string value. To have only Double-Take service messages displayed, specify EN for the string value.	false	false
Last Sequenece Number	Int32	Specify a sequence number to be the starting point to retrieve the log messages.	false	false
LastTime Stamp	DateTime Offset	Specify a date and time stamp to be the starting point to retrieve the log messages. Specify the date in mm/dd/yyyy format. Specify the time in hh:mm:ss format with AM or PM. You can specify a time zone offset, for example, -04:00. If you do not specify a time zone offset, the time zone of the machine you are running from will be used. If you do not specify a time, 12:00:00 AM will be used.	false	false
ChunkSize	Int32	Specify the number of entries that will be returned at one time. The default number of entries is 1024.	false	false

Outputs

LogMessage [ ] on page 283

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Get-DtLogMessage -ServiceHost $DtServerObjectAlpha -source MS -LastTimeStamp "01/15/2018 05:19:00 PM" -ChunkSize 25

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the Double-Take Management Service log messages after 5:19pm on January 15, 2018 are displayed in groups of 25. The connections for the server object are then closed.

Get-DtOnlineActivationRequest

Returns the server information that is required to activate a license

Syntax

Get-DtOnlineActivationRequest [-ServiceHost] <Server> [<CommonParameters>]

Detailed Description

This cmdlet returns server information, unique to a specific, single server, that is required to activate a Carbonite license. The server must already have a license key on the server in order to get the server information.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false

Outputs

ServerActivationInformation on page 322

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Get-DtOnlineActivationRequest -ServiceHost $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The server information for the online activation process is returned. The connections for the server object are then closed.

Get-DtOption

Returns job or server value

Syntax

Get-DtOption [-ServiceHost] <Server> [[-Name] <String[]>] [<CommonParameters>]

Detailed Description

This cmdlet returns the value of the specific job or server option from the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
Name	String	Specify the name of the job or server option. See the *Reference Guide* for details on each job and server option. Specify multiple strings in an array using the format @ (string1, string2).	false	false

Outputs

Hashtable

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Get-DtOption -ServiceHost $DtServerObjectAlpha -Name MirrorChunkSize
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the value of the server option called MirrorChunkSize is returned from the server. The connections for the server object are then closed.

Get-DtPathBlocking

Returns the blocked paths

Syntax

Get-DtPathBlocking [-ServiceHost] <Server> [<CommonParameters>]

Detailed Description

This cmdlet returns the paths that are blocked on the specified target server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false

Outputs

PathBlocking on page 297

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Get-DtPathBlocking -ServiceHost $DtServerObjectAlpha
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the paths that are blocked on the server are returned. The connections for the server object are then closed.

Get-DtPhysicalItem

Returns files system information

Syntax

This cmdlet returns file system information for the specified server. A physical item can be used to specify a specific file, folder, or volume to return file system information for.

Detailed Description

Get-DtPhysicalItem [-ServiceHost] <Server> [-Ref <PhysicalItem>] [<CommonParameters>]

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
Ref	PhysicalItem on page 299	Specify an object returned from a previous Get-DtPhysicalItem call.	false	false

Outputs

PhysicalItem [ ] on page 299

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtVolumes = Get-DtPhysicalItem -ServiceHost $DtServerObjectAlpha

$DtVolume1Root = Get-DtPhysicalItem -ServiceHost $DtServerObjectAlpha -Ref $DtVolumes[0]

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the volumes on the server are stored in the variable DtVolumes. Finally, the files and folders at the root of the first volume in DtVolumes is stored in the variable DtVolume1Root. The connections for the server object are then closed.

Get-DtProductInfo

Returns Carbonite product information

Syntax

Get-DtProductInfo [-ServiceHost] <Server> [<CommonParameters>]

Detailed Description

This cmdlet returns Carbonite product information for the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false

Outputs

ProductInfo on page 305

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Get-DtProductInfo -ServiceHost $DtServerObjectAlpha
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the Carbonite product information for the server is returned. The connections for the server object are then closed.

Get-DtQualificationResults

Returns the qualification results

Syntax

Get-DtQualificationResults [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Get-DtQualificationResults [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet returns the qualification results for the specified job type. You may want to use these results for job options when editing a job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	false

Outputs

JobQualificationResults on page 275

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Get-DtQualificationResults -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id
Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable

DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The qualification results for the job are then returned. The connections for the server object are then closed.

Get-DtRecommendedFailbackOptions

Returns the recommended failback options

Syntax

Get-DtRecommendedFailbackOptions [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Get-DtRecommendedFailbackOptions [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet returns the recommended failback options for the specified job on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	false

Outputs

RecommendedFailbackOptions on page 309

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password
$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}
Get-DtRecommendedFailbackOptions -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id
Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Then the

recommended failback options for the specified job and server are returned. The connections for the server object are then closed.

Get-DtRecommendedFailoverOptions

Returns the recommended failover options

Syntax

Get-DtRecommendedFailoverOptions [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Get-DtRecommendedFailoverOptions [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters

Detailed Description

This cmdlet returns the recommended failover options for the specified job on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	false	false

Outputs

RecommendedFailoverOptions on page 310

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Get-DtRecommendedFailoverOptions -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id
Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Then the

recommended failover options for the specified job and server are returned. The connections for the server object are then closed.

Get-DtRecommendedJobOptions

Returns recommended job options

Syntax

Get-DtRecommendedJobOptions [-ServiceHost] <Server> [-Source] <Server> [-JobType] <String> [-Workload] <Workload> [-OtherServers <Server[]>] [<CommonParameters>]

Detailed Description

This cmdlet returns the recommended job options for the specified job type.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
Source	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103.	true	false

JobType	String	This value is the job type name.
		• **Availability for Windows jobs**
		  • **FilesAndFolders**—Files and folders
		  • **ClusterAwareFilesAndFolders**—Cluster-aware files and folders
		  • **SQL**—SQL
		  • **ClusterAwareSql**—Cluster-aware SQL
		  • **FullServerFailover**—Full server
		  • **VRA**—Full server to ESX or full server to Hyper-V
		• **Availability for Linux jobs**
		  • **LinuxFullServerFailover**—Full server
		  • **Lvra**—Full server to ESX
		• **Migrate for Windows jobs**
		  • **MoveDataOnlyMigration**—Files and folders migration
		  • **MoveServerMigration**—Full server migration
		  • **VraMove**—Full server to ESX migration or full server to Hyper-V migration
		• **Migrate for Linux jobs**
		  • **LinuxMoveServerMigration**—Full server
			true	false

Name	Type	Description	Required	Pipeline Input
		migration • **MoveLvra**—Full server to ESX migration • **Other jobs** • **Diagnostics**—Throughput Diagnostic Utility		
Workload	Workload on page 365	Specify the workload object returned from the Get-DtWorkload cmdlet. See Get-DtWorkload on page 82.	true	false
Other Servers	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. Specify multiple server objects in an array using the format @($server1, $server2).	false	false

Outputs

RecommendedJobOptions on page 311

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName FilesAndFolders

$DtWorkload = Get-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid

$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtServerObjectBeta -Source $DtServerObjectAlpha -JobType FilesAndFolders -Workload $DtWorkload

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The script then creates a workload on the server for a files and folders job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. The workload definition for the workload type and the server is then stored in the DtWorkload variable. The recommended job options for the servers and the workload type are then stored in the variable DtJobOptions. The connections for the server object are then closed.

Get-DtRecommendedPathTransform

Returns the recommended mappings

Syntax

This cmdlet returns the recommended mapping between the location of the data on the source and the location of the replica data on the target for the specific type of workload .

Detailed Description

Get-DtRecommendedPathTransform [-ServiceHost] <Server> [-WorkloadId] <Guid> [-BasePath <String>] [<CommonParameters>]

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your source server.	true	false
WorkloadId	Guid on page 267	Specify the workload GUID returned from the New-DtWorkload cmdlet using the workload type name parameter. See New-DtWorkload on page 110.	true	false
BasePath	String	Specify the location on the target where the replica of the source data will be stored. By default, the replica source data will be stored in the same directory structure on the target, in a one-to-one configuration.	false	false

Outputs

PathTransformation [ ] on page 298

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha-WorkloadTypeName FullServerFailover

Get-DtRecommendedPathTransform -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The script then creates a workload on the server for a full sever job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. The recommended mapping of the data on the source and the replica data on the target is then returned. The connections for the server object are then closed.

Get-DtRecommendedRestoreOptions

Returns the recommended restoration options

Syntax

Get-DtRecommendedRestoreOptions [-ServiceHost] <Server> [-JobId] <Guid> [-RestoreTarget <Server>] [-RequestCanClearRestoreRequired] [<CommonParameters>]

Get-DtRecommendedRestoreOptions [-ServiceHost] <Server> -JobInfo <JobInfo> [-RestoreTarget <Server>] [-RequestCanClearRestoreRequired] [<CommonParameters>]

Detailed Description

Returns the recommended restoration options for the specified job on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Restore Target	Server on page 320	Specify the server you want to restore to. If you do not specify a server, the original source server will be used.	false	false
Request CanClear Restore Required	Switch Parameter	Sets RecommendedRestoreOptions.CanClearRestoreRequired on the RecommendedRestoreOptions object based on whether it is supported by the current job state. You still need to set ClearRestoreRequired on RestoreOptions to clear the restore required state of a job.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	false

Outputs

RecommendedRestoreOptions on page 312

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Get-DtRecommendedRestoreOptions -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -
RestoreTarget $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Then the recommended restore options for the specified job and server are returned. The connections for the server object are then closed.

If you want to clear the restore required flag, save your recommended restore options to a variable and then set ClearRestoreRequired to true.

```
$DtRecommendedRestoreOptions = Get-DtRecommendedRestoreOptions -ServiceHost
$DtServerObjectBeta -JobId $DtJobForAlpha.Id -RestoreTarget $DtServerObjectAlpha -
RequestCanClearRestoreRequired

$DtRestoreOptions = $DtRecommendedRestoreOptions.RestoreOptions

$DtRestoeOptions.ClearRestoreRequired = $true;
```

Get-DtRepairJobOptionsStatus

Returns the details and status of a repair

Syntax

Get-DtRepairJobOptionsStatus [-ServiceHost] <Server> [-Token] <ActivityToken> [<CommonParameters>]

Detailed Description

This cmdlet returns the details and status of the repair performed by the Repair-DtJobOptions cmdlet. See Repair-DtJobOptions on page 120.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
Token	ActivityToken on page 213	Specify the repair action object returned from the Repair-DtJobOptions cmdlet. See Repair-DtJobOptions on page 120.	true	false

Outputs

RepairStatus on page 313

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtValidation = Confirm-DtJobOptions -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -
JobOptions $DtJob.Options

$DtStatus = Get-DtVerificationStatus -ServiceHost $DtServerObjectBeta -Token $DtValidation

$DtRepair = Repair-DtJobOptions -ServiceHost $DtTarget -JobId $DtJob.Id -JobOptions $DtJob.Options -
Step $DtStatus.Steps

$DtRepairStatus = Get-DtRepairJobOptionsStatus -ServiceHost $DtServerObjectBeta -Token $DtRepair

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The job options

used by the job are confirmed, and the validation result is stored in DtValidation. The details of the validation are stored in the variable DtStatus. Ideally you should script this with a loop so you can wait until the verification status is complete before continuing with your script. Those items that can automatically be fixed are corrected. If the job options were modified in order to fix an issues, the updated job options are now contained in the variable $DtRepair. The details and status of the repair are stored in the variable DtRepairStatus. The connections for the server object are then closed.

Get-DtScriptCredentials

Returns credentials

Syntax

Get-DtScriptCredentials [-ServiceHost] <Server> [<CommonParameters>]

Detailed Description

This cmdlet returns the credentials that Carbonite is currently using to run scripts on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Get-DtScriptCredentials -ServiceHost $DtServerObjectAlpha
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the credentials that Carbonite is using to run scripts on this server are returned. The connections for the server object are then closed.

Get-DtServerInfo

Returns server information

Syntax

Get-DtServerInfo [-ServiceHost] <Server> [<CommonParameters>]

Detailed Description

This cmdlet returns server configuration information for the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false

Outputs

ServerInfo on page 323

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Get-DtServerInfo -ServiceHost $DtServerObjectAlpha
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then server configuration information for the server is returned. The connections for the server object are then closed.

Get-DtSnapshot

Returns snapshots

Syntax

Get-DtSnapshot [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Get-DtSnapshot [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet returns the available Carbonite snapshots for the specified job .

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

SnapshotEntry on page 329

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password
$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}
Get-DtSnapshot -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id
Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable

DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The snapshots available for the job are returned. The connections for the server object are then closed.

Get-DtSourceQueueSnapshot

Gets a coordinated snapshot

Syntax

Get-DtSourceQueueSnapshot [-ServiceHost] <Server> [-CorrelationId] <Guid> [-JobId] <Guid> [<CommonParameters>]

Detailed Description

This cmdlet gets a coordinated snapshot for one job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
CorrelationID	Guid	Specify the correlation ID used to create the coordinated snapshot. See Checkpoint-DtConnectionSourceQueue on page 22.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false

Outputs

SourceQueueSnapshotEntry on page 331

Examples

```
$CorId=New-Guid
$Source1Uri = "dtms://112.42.7.63:6325/"
$Source2Uri = "dtms://112.42.7.71:6325/"
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password
$DtJob1 = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object { $_.SourceHostUri -eq $Source1Uri}
$DtJob2 = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object { $_.SourceHostUri -eq $Source2Uri}
$DtJobsArray = New-Object "System.Collections.Generic.List[Guid]"
$DtJobsArray.Clear()
```

```
$DtJobsArray.Add($DtJob1.Id)

$DtJobsArray.Add($DtJob2.Id)

Checkpoint-DtConnectionSourceQueue -ServiceHost $DtServerObjectBeta -CorrelationId $CorId -JobIds
$DtJobsArray.ToArray()

Get-DtSourceQueueSnapshot -ServiceHost $DtServerObjectBeta -CorrelationId $CorId -JobIds $DtJob1.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

Several variables are set including a unique correlation ID for the snapshots that will be taken as well as URI identifiers for the source servers of existing jobs. A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The jobs are retrieved from DtServerObjectBeta, and the job information is inserted into unique variables. And array of job IDs is then created. A coordinated snapshot is taken using the specified correlation ID and the jobs stored in the array. Only those jobs will have coordinated snapshots. Then the coordinated snapshot for the specified correlation ID and job ID is returned. The connections for the server object are then closed.

Get-DtSourceQueueSnapshots

Gets all coordinated snapshots

Syntax

Get-DtSourceQueueSnapshot [-ServiceHost] <Server> [-CorrelationId] <Guid> [<CommonParameters>]

Detailed Description

This cmdlet gets all coordinated snapshot for one correlation ID.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
CorrelationID	Guid	Specify the correlation ID used to create the coordinated snapshot. See Checkpoint-DtConnectionSourceQueue on page 22.	true	false

Outputs

SourceQueueSnapshotEntry on page 331 [ ]

Examples

```
$CorId=New-Guid
$Source1Uri = "dtms://112.42.7.63:6325/"
$Source2Uri = "dtms://112.42.7.71:6325/"
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password
$DtJob1 = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object { $_.SourceHostUri -eq $Source1Uri}
$DtJob2 = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object { $_.SourceHostUri -eq $Source2Uri}
$DtJobsArray = New-Object "System.Collections.Generic.List[Guid]"
$DtJobsArray.Clear()
$DtJobsArray.Add($DtJob1.Id)
$DtJobsArray.Add($DtJob2.Id)
Checkpoint-DtConnectionSourceQueue -ServiceHost $DtServerObjectBeta -CorrelationId $CorId -JobIds
$DtJobsArray.ToArray()
```

```
Get-DtSourceQueueSnapshots -ServiceHost $DtServerObjectBeta -CorrelationId $CorId

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

Several variables are set including a unique correlation ID for the snapshots that will be taken as well as URI identifiers for the source servers of existing jobs. A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The jobs are retrieved from DtServerObjectBeta, and the job information is inserted into unique variables. And array of job IDs is then created. A coordinated snapshot is taken using the specified correlation ID and the jobs stored in the array. Only those jobs will have coordinated snapshots. Then the coordinated snapshots for the specified correlation ID are returned. The connections for the server object are then closed.

Get-DtUvraRecommendedFailoverOptions

Returns the recommended failover options

Syntax

Get-DtUvraRecommendedFailoverOptions [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Get-DtUvraRecommendedFailoverOptions [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet returns the recommended failover options for the specified full server to ESX appliance job on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtUvraServer cmdlet. See New-DtUvraServer on page 108. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	false	false

Outputs

FailoverOptions on page 257

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName root -Password password

$DtApplianceObject = New-DtUvraServer -Name beta -UserName root -Password password -Port 6325

$DtJobForAlpha = Get-DtJob -ServiceHost $DtApplianceObject | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Get-DtUvraRecommendedFailoverOptions -ServiceHost $DtApplianceObject -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtApplianceObject
```

```
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the root and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. A server object is created for the appliance beta using port 6325 and the root and password credentials. It assigns the server object to the variable called DtApplianceObject. The job (s) are retrieved from DtApplianceObject, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Then the recommended failover options for the specified full server to ESX appliance job and server are returned. The connections for the server object are then closed.

Get-DtUvraRecommendedRemoveOptions

Returns recommended removal options

Syntax

Get-DtUvraRecommendedRemoveOptions [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Get-DtUvraRecommendedRemoveOptions [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet returns the recommended removal options when deleting the specified full server to ESX appliance job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtUvraServer cmdlet. See New-DtUvraServer on page 108. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	false	false

Outputs

DeleteOptions on page 242

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName root -Password password

$DtApplianceObject = New-DtUvraServer -Name beta -UserName root -Password password -Port 6325

$DtJobForAlpha = Get-DtJob -ServiceHost $DtApplianceObject | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Get-DtUvraRecommendedRemoveOptions -ServiceHost $DtApplianceObject -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtApplianceObject
```

```
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the root and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. A server object is created for the appliance beta using port 6325 and the root and password credentials. It assigns the server object to the variable called DtApplianceObject. The job (s) are retrieved from DtApplianceObject, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Then the recommended remove options for the specified full server to ESX appliance job and server are returned. The connections for the server object are then closed.

Get-DtVerificationStatus

Returns the validation details and status

Syntax

Get-DtVerificationStatus [-ServiceHost] <Server> [-Token] <ActivityToken> [<CommonParameters>]

Detailed Description

This cmdlet returns the details and status of the validation performed by the Confirm-DtJobOptions cmdlet. See Confirm-DtJobOptions on page 25. Do not confuse this process with the verification process that confirms if the data between the source and target are synchronized.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
Token	ActivityToken on page 213	Specify the confirm action object returned from the Confirm-DtJobOption cmdlet. See Confirm-DtJobOptions on page 25.	true	false

Outputs

VerificationStatus on page 345

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtValidation = Confirm-DtJobOptions -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -
JobOptions $DtJob.Options

$DtStatus = Get-DtVerificationStatus -ServiceHost $DtServerObjectBeta -Token $DtValidation

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The job options used by the job are confirmed, and the validation result is stored in DtValidation. The details of the validation are stored in the variable DtStatus. Ideally you should script this with a loop so you can wait until the verification status is complete before continuing with your script. The connections for the server object are then closed.

Get-DtWorkload

Returns the workload definition

Syntax

Get-DtWorkload [-ServiceHost] <Server> [-WorkloadId] <Guid> [<CommonParameters>]

Detailed Description

This cmdlet returns an object that represents the workload definition, including the workload type name, any physical rules, and any logical rules. This object is used in job cmdlets.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your source server.	true	false
WorkloadId	Guid on page 267	Specify the workload GUID returned from the New-DtWorkload cmdlet using the workload type name parameter. See New-DtWorkload on page 110.	true	false

Outputs

Workload on page 365

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha-WorkloadTypeName FullServerFailover

$DtWorkload = Get-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The script then creates a workload on the server for a full sever job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. The workload definition for the workload type and the server is then stored in the DtWorkload variable. The connections for the server object are then closed.

Get-DtWorkloadPhysicalItem

Returns physical items

Syntax

Get-DtWorkloadPhysicalItem [-ServiceHost] <Server> [-WorkloadId] <Guid> [-RefItem <PhysicalItem>] [<CommonParameters>]

Detailed Description

This cmdlet returns the physical items available for the specified workload on the specified server

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your source server.	true	false
WorkloadId	Guid on page 267	Specify the workload GUID returned from the New-DtWorkload cmdlet using the workload type name parameter. See New-DtWorkload on page 110.	true	false
Ref	PhysicalItem on page 299	Specify an object returned from a previous Get-DtWorkloadPhysicalItem call.	false	false

Outputs

PhysicalItem on page 299

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName FilesAndFolders

Get-DtWorkloadPhysicalItem -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The script then creates a workload on the server for a files and folders job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. Finally, the physical items available for the workload on the server are returned. The connections for the server object are then closed.

Get-DtWorkloadType

Returns the workload types

Syntax

Get-DtWorkloadType [-ServiceHost] <Server> [<CommonParameters>]

Detailed Description

This cmdlet returns the types of workloads that are supported on the specified server. The supported workload types are based on the Carbonite license keys on the server, the applications on the server, the server configuration (like standalone or cluster), and so on.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your source server.	true	false

Outputs

WorkloadType [ ] on page 368

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Get-DtWorkloadType -ServiceHost $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then it returns the workload types that are supported on the server. The connections for the server object are then closed.

Install-DoubleTake

Installs Carbonite

Syntax

Install-DoubleTake [-RemoteServer] <Server> -ActivationCode <String[]> [-DiskQueueFolder <String>] [-DiskQueueLimit <Int32>] [-DotNetPackagePath <String>] [-InstallationFolder <String>] [-MaxMemoryUsage <Int32>] [-MinFreeDiskSpace <Int32>] [-PackageBaseFolder <String>] [-SimultaneousFilePushLimit <Int32>] [-TempFolder <String>] [-X64PackageFolder <String>] [-X86PackageFolder <String>] [-LinuxPackageFolder <String>] [-AsJob] [<CommonParameters>]

Install-DoubleTake [-RemoteServer] <Server> -ActivationCode <String[]> -Schedule <DateTime> [-DiskQueueFolder <String>] [-DiskQueueLimit <Int32>] [-DotNetPackagePath <String>] [-InstallationFolder <String>] [-MaxMemoryUsage <Int32>] [-MinFreeDiskSpace <Int32>] [-NoReboot] [-PackageBaseFolder <String>] [-SimultaneousFilePushLimit <Int32>] [-TempFolder <String>] [-X64PackageFolder <String>] [-X86PackageFolder <String>] [-AsJob] [<CommonParameters>]

Detailed Description

This cmdlet installs Carbonite on the specified server. The first syntax allows you to install Carbonite immediately. The second syntax allows you to schedule the installation.

Parameters

Name	Type	Description	Required	Pipeline Input
Remote Server	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103.	true	false
Activation Code	String	You have the following options. • **Use this parameter and enter a key**—If you use this parameter, you can specify a 24-character, alpha-numeric license key which applies the appropriate Carbonite license to your Carbonite server. If there is an existing key, it will be overwritten. Specify multiple keys in an array using the format @(code1, code2). • **Use this parameter and enter None**—If you use this parameter, you can specify the keyword **None**. If there is an existing key, it will be removed and no key written. • **Do not use this parameter**—if you do not use this parameter, existing keys will be maintained as is.	false	false
DiskQueue Folder	String	Specify the location where you want to store the Carbonite disk queue on each server. The default is \Program Files\Carbonite\Replication.	false	false

Name	Type	Description	Required	Pipeline Input
DiskQueue Limit	Int32	Specify a fixed amount of disk space, in MB, in the specified DiskQueueFolder that can be used for Carbonite disk queuing. When the disk space limit is reached, Carbonite will automatically begin the auto-disconnect process. By default, an unlimited amount of disk queuing will be allowed.	false	false
DotNet Package Path	String	If your servers do not have Microsoft .NET version 4.5.1, specify the location of the setup file (on the local machine) that will be used to install it. This option is for older versions of Windows. Generally, newer versions of Windows will have .NET already installed.	false	false
Installation Folder	String	Specify the location where you want to install Carbonite on the server. The default is \Program Files\Carbonite\Replication.	false	false
Max Memory Usage	Int32	Specify the maximum amount of memory, in MB, that can be used for Carbonite processing. The default will depend on your operating system and hardware. For complete details on memory usage, see the *User's Guide*.	false	false
MinFree DiskSpace	Int32	This is the minimum amount of disk space in the specified DiskQueueFolder that must be available at all times. This amount should be less than the amount of physical disk space minus the disk size specified for DiskQueueLimit. The default is 50 MB.	false	false
Package Base Folder	String	Specifies the locations of the setup files (on the local machine or a UNC path) that will be used to install on both 32-bit and 64-bit servers. By default, these are in the i386\ and \x64 subdirectories where you installed Carbonite.	false	false
Simultaneous FilePush Limit	Int32	Specify the number of files that can simultaneously be pushed to the machine you are installing on. The default is 5.	false	false
Temp Folder	String	Specify a temporary location (on the server where you are installing Carbonite) where the installation files will be copied and run. The default is \Temp. You need approximately 130 MB of space in the specified location.	false	false
X64 Package Folder	String	Specify the location of the setup file (on the local machine or a UNC path) that will be used to install on 64-bit Windows servers. By default, this is in the \x64 subdirectory where you installed Carbonite.	false	false

Name	Type	Description	Required	Pipeline Input
X86 Package Folder	String	Specify the location of the setup file (on the local machine or a UNC path) that will be used to install on 32-bit Windows servers. By default, this is in the \i386 subdirectory where you installed Carbonite.	false	false
Linux Package Folder	String	Specify the location of the .rpm or .deb installation files (on the local machine or a UNC path) that will be used to install on Linux servers.	false	false
AsJob	Switch Parameter	Specify if you want the installation to occur asynchronously in the background, returning the PowerShell command immediately. You can get the status of each installation using the Windows PowerShell Get-Job command. Without this parameter, each push installation specified will be executed synchronously and the current activity of the current installation will be displayed.	false	false
Schedule	DateTime	Specify a date and time to complete the installation. Specify the date in mm/dd/yyyy format. Specify the time in hh:mm:ss format with AM or PM. You can specify a time zone offset, for example, -04:00. If you do not specify a time zone offset, the time zone of the machine you are running from will be used. If you do not specify a time, 12:00:00 AM will be used.	true	false
NoReboot	Switch Parameter	Specify if you do not want the server to reboot after the installation, even if a reboot is required.	false	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Install-DoubleTake -RemoteServer $DtServerObjectAlpha -ActivationCode 1234567890abcdefghij1234 -x64PackageFolder "C:\Program Files\Carbonite\Replication\x64"

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then Carbonite is installed to the server using the license key 1234567890abcdefghij1234, using the setup.exe installation file stored locally at C:\Program Files\Carbonite\Replication\x64. The connections for the server object are then closed.

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Install-DoubleTake -RemoteServer $DtServerObjectAlpha -ActivationCode 1234567890abcdefghij1234 -LinuxPackageFolder "C:\Program Files\Carbonite\Replication\Linux"

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then Carbonite is installed to the server using the license key 1234567890abcdefghij1234, using the .rpm or .deb installation files stored locally at C:\Program Files\Carbonite\Replication\Linux. The connections for the server object are then closed.

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Install-DoubleTake -RemoteServer $DtServerObjectAlpha -ActivationCode 1234567890abcdefghij1234 -LinuxPackageFolder "\\DTAppliance\installers"

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then Carbonite is installed to the server using the license key 1234567890abcdefghij1234, using the .rpm or .deb installation files stored on a deployed Carbonite Linux appliance at \\DTAppliance\installers. The connections for the server object are then closed.

Install-DtVmwareCertificate

Installs VMware security certificate

Syntax

Install-DtVmwareCertificate [-ServiceHost] <Server> [-Certificate] <X509Certificate> [<CommonParameters>]

Detailed Description

This cmdlet installs the specified X.509 security certificate on the specified server

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
Certificate	$Variable [1]	The certificate from the Test-DtVmwareCertificatePolicy on page 192 false return must be installed. The certificate is the second item in the returned array.	true	false

Outputs

None

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$TestResult = Test-DtVmwareCertificatePolicy -ServiceHost $DtServerObjectBeta -VmwareServer 112.47.15.6

Set-DtVmwareCertificatePolicy -ServiceHost $DtServerObjectBeta -VmwareServer 112.47.15.6 -Policy AllowAll

Install-DtVmwareCertificate -ServiceHost $DtServerObjectBeta -Certificate $TestResult[1]

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The server $DTServerObjectBeta is tested to see if the current policy or any valid certificates will allow a connection to the specified VMware server. In this example, assume the return is false. The VMware certificate policy on the server $DTServerObjectBeta is then set to allow all certificates to be installed. The certificate from the Test-DtVmwareCertificatePolicy false return is then installed on the server $DTServerObjectBeta. The connections for the server object are then closed.

Invoke-DtAddShares

Adds shares to the target

Syntax

Invoke-DtAddShares [-ServiceHost] <Server> [-JobXmlPath] <String> [[-SharePathFilter] <String>] [<CommonParameters>]

Detailed Description

This cmdlet takes the drive share configuration that was gathered during Carbonite mirroring and replication and applies it to the target server. This process is independent of the failover process.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobXml Path	String	Specify the path and file name to the Carbonite .xml configuration file for the job that contains the protected shares. By default, this location is \Program Files\Carbonite\Replication\Service\Data. The file name will be JobGuid.xml where Guid is the unique identifier assigned to the job. You can find this ID by using Get-DtJob on page 43.	true	false
SharePath Filter	String	Creates shares from the specified path and any subdirectories of this path	false	false

Outputs

None

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobInfo = Get-DtJob -ServiceHost $DtServerObjectBeta

$JobId = $DtJobInfo.Id

$PathToJobFile = "C:\Program Files\Carbonite\Replication\Service\Data\"

$JobFileExtension = ".xml"

$JobFile = $PathToJobFile + "Job" + $JobId + $JobFileExtension

Invoke-DtAddShares –ServiceHost $DtServerObjectBeta –JobXmlPath $JobFile
```

```
Disconnect-DtServer –ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. All job information for all of the jobs on the server beta are stored in the variable DtJobInfo. This type of usage is common when the jobs were created in the past or if you did not store or do not know a job's ID. In this example, assume there is only one job on the server. See Get-DtJob on page 43 for examples on how to get specific job information when there are multiple jobs on one server. Job id from the stored job information is stored in the variable JobId. The path to the location of the job file is stored in the variable PathToJobFile. The extension of the job file is stored in the variable JobFileExtension. The variables are appended, together with the word Job, to create the variable JobFile. For example, the JobFile variable might end up being C:\Program Files\Carbonite\Replication\Service\Data\Job87667a0d-4516-4182-ab8f-13f6fd15dd92.xml. Shares are then added on the server DtServerObjectBeta using the job configuration file stored in JobFile. The connections for the server object are then closed.

Invoke-DtQueueTask

Queues tasks

Syntax

Invoke-DtQueueTask [-ServiceHost] <Server> [-JobId] <Guid> [-OnQueue <TaskParameters>] [-OnTransmit <TaskParameters>] [-OnReceive <TaskParameters>] [-OnExecute <TaskParameters>] [-InteractWithDesktop] [-Timeout <TimeSpan>] [-ConnectionId <Guid>] [<CommonParameters>]

Invoke-DtQueueTask [-ServiceHost] <Server> -JobInfo <JobInfo> [-OnQueue <TaskParameters>] [-OnTransmit <TaskParameters>] [-OnReceive <TaskParameters>] [-OnExecute <TaskParameters>] [-InteractWithDesktop] [-Timeout <TimeSpan>] [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet queues tasks inline with replication data. Keep the following in mind when using this cmdlet.

- Any combination of one or more execution points can be used with the same Invoke-DtQueueTask cmdlet.
- All script processing messages, including errors, can be viewed in the Carbonite log and the Windows Event log.
- If your source is in a restore required state (after a failover), any task placed on the queue will be executed immediately. Use caution when submitting tasks while in this state so that the target does not get inadvertently updated.
- If a task is submitted after replication is stopped, the task will be executed immediately.
- A task may be discarded if all jobs to a target are manually stopped, if replication is stopped to a target, or if an auto-disconnect occurs.
- If you disable task command processing while tasks are in queue, those tasks will not be executed.
- The user submitting the task command must be a member of the Double-Take Admin security group on both the source and target and the Double-Take service must have proper privileges to access the files or run the commands specified in the task.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
OnQueue	TaskParameters on page 339	Execute the specified task on the source machine as soon as the source receives and queues the	false	false

Name	Type	Description	Required	Pipeline Input
		task. During heavy replication, there may be a delay while the task is queued inline with the replication operations. Define the task parameters by using New-DtTaskParameters on page 105.		
OnTransmit	TaskParameters on page 339	Execute the specified task on the source machine just before the source transmits the task to the target. Define the task parameters by using New-DtTaskParameters on page 105.	false	false
OnReceive	TaskParameters on page 339	Execute the specified task on the target machine as soon as the target receives and queues the task. Define the task parameters by using New-DtTaskParameters on page 105.	false	false
OnExecute	TaskParameters on page 339	Execute the specified task on the target when the target processes the task from the queue. Since the task is not executed until it is processed, if the target is paused, the task will be held in queue. Define the task parameters by using New-DtTaskParameters on page 105.	false	false
Interact With Desktop	SwitchParameter	Tasks interact with the desktop and, therefore, display on screen and run in the foreground. If you do not use this option, tasks do not interact with the desktop and will be run in the background.	false	false
Timeout	TimeSpan	Specify the length of time, in timespan format, to wait for tasks to complete. For example, 0.01:30:00 would wait for one hour and thirty minutes. If you set the timespan to zero (0.00:00:00), there is no timeout delay and the next operation is immediately processed. If you do not specify a timeout parameter, the timeout will default to forever.	false	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtScript = New-DtTaskParameters -ScriptPath "C:\PathDir\ScriptName" -Arguments "arg1 arg2"

$DtPsScript = New-DtTaskParameters -ScriptPath
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -Arguments "-File
""C:\PathDir\Script.ps1"" ""-Arg1 argument1_info -Arg2 argument2_info"" -ExecutionPolicy RemoteSigned"

Invoke-DtQueueTask -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -OnReceive $DtScript -
OnExecute $DtPsScript

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The script called ScriptName, located in C:\PathDir, along with two arguments, is stored in the variable DtScript. The script to launch PowerShell and run the script called Script.ps1, located in C:\PathDir, along with two arguments and the ExecutionPolicy parameter, is stored in the variable DtPsScript. Finally, the script stored in DtScript is executed when the target receives and queues the task and the script stored in DtPsScript is executed when the target processes the task from the queue. The connections for the server object are then closed.

Invoke-DtRemoveShares

Removes shares from the target

Syntax

Invoke-DtRemoveShares [-ServiceHost] <Server> [-JobXmlPath] <String> [[-SharePathFilter] <String>] [<CommonParameters>]

Detailed Description

This cmdlet takes the drive share configuration that was gathered during Carbonite mirroring and replication and removes it from the target server. This process is independent of the failback process.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobXml Path	String	Specify the path and file name to the Carbonite .xml configuration file for the job that contains the protected shares. By default, this location is \Program Files\Carbonite\Replication\Service\Data. The file name will be JobGuid.xml where Guid is the unique identifier assigned to the job. You can find this ID by using Get-DtJob on page 43.	true	false
SharePath Filter	String	Removes shares from the specified path and any subdirectories of this path	false	false

Outputs

None

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobInfo = Get-DtJob -ServiceHost $DtServerObjectBeta

$JobId = DtJobInfo.Id

$PathToJobFile = "C:\Program Files\Carbonite\Replication\Service\Data\"

$JobFileExtension = ".xml"

$JobFile = $PathToJobFile + "Job" + $JobId + JobFileExtension

Invoke-DtRemoveShares –ServiceHost $DtServerObjectBeta –JobXmlPath $JobFile

Disconnect-DtServer – ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. All job information for all of the jobs on the server beta are stored in the variable DtJobInfo. This type of usage is common when the jobs were created in the past or if you did not store or do not know a job's ID. In this example, assume there is only one job on the server. See Get-DtJob on page 43 for examples on how to get specific job information when there are multiple jobs on one server. Job id from the stored job information is stored in the variable JobId. The path to the location of the job file is stored in the variable PathToJobFile. The extension of the job file is stored in the variable JobFileExtension. The variables are appended, together with the word Job, to create the variable JobFile. For example, the JobFile variable might end up being C:\Program Files\Carbonite\Replication\Service\Data\Job87667a0d-4516-4182-ab8f-13f6fd15dd92.xml. Shares are then removed on the server DtServerObjectBeta using the job configuration file stored in JobFile. The connections for the server object are then closed.

Merge-DtConsoleServerData

Applies Carbonite Replication Console server data from a file

Syntax

Merge-DtConsoleServerData [-ImportFilePath] <String> [-Replace] [<CommonParameters>]

Detailed Description

This cmdlet applies the server information from a server data file generated from the Save-DtConsoleServerData cmdlet. See Save-DtConsoleServerData on page 133. The server information from the file will be applied to the user profile of the user that is currently logged into the machine. If you have multiple administrators sharing the same Carbonite Replication Console installation, each administrator will need to be logged in to apply console server data to their user profile. You must close the Carbonite Replication Console to apply the server data file.

Parameters

Name	Type	Description	Required	Pipeline Input
Import FilePath	String	Specify the path and file name of the .xml console data file that was generated from the Save-DtConsoleServerData cmdlet. See Save-DtConsoleServerData on page 133.	true	false
Replace	Switch Parameter	Remove the servers from the console that are not present in the import file.	false	false

Outputs

None

Examples

```
Merge-DtConsoleServerData -ImportFilePath "C:\DtAdmin1_ServerData.xml"
```

The Carbonite Replication Console server data from the file DtAdmin1_ServerData.xml is applied to the user profile for the user that is currently logged in.

New-DtFilesAndFoldersJob

Creates a files and folders job

Syntax

New-DtFilesAndFoldersJob [-ServiceHost] <Server> [-Source] <Server> [-Path] <String> [[-TargetPath] <String>] [-Name <String>] [-JobOptions <JobOptions>] [<CommonParameters>]

Detailed Description

This cmdlet creates a files and folders job on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
Source	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103.	true	false
Path	String	Specify the path on the source that contains the data that you want to protect	true	false
TargetPath	String	Specify the path on the target where you want to store the replica data from the source. By default, a one-to-one mapping will be used on the target, which means the replica source data will be stored in the same directory structure on the target.	false	false
Name	String	Specify the name of the job.	false	false
JobOptions	JobOptions on page 273	Specify the JobOptions returned from the Get-DtRecommendedJobOptions cmdlet. See Get-DtRecommendedJobOptions on page 62.	false	false

Outputs

Guid on page 267

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName FilesAndFolders
```

```
$DtWorkload = Get-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid

$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtServerObjectBeta -Source
$DtServerObjectAlpha -JobType FilesAndFolders -Workload $DtWorkload

New-DtFilesAndFoldersJob -ServiceHost $DtServerObjectBeta -Source $DtServerObjectAlpha -Path
"C:\Data" -TargetPath "C:\Alpha\C" -Name "Alpha to Beta" -JobOptions $DtJobOptions.JobOptions

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The script then creates a workload on the server for a files and folders job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. The workload definition for the workload type and the server is then stored in the DtWorkload variable. The recommended job options for the servers and the workload type are then stored in the variable DtJobOptions. A new files and folders job is created using the servers and the job options. The connections for the server object are then closed.

New-DtJob

Creates a job

Syntax

New-DtJob [-ServiceHost] <Server> [-Source] <Server> [-JobType] <String> [-JobOptions] <JobOptions> [[-OtherServers] <Server[]>] [<CommonParameters>]

New-DtJob [-ServiceHost] <Server> [-CreateOptions] <CreateOptions> [<CommonParameters>]

Detailed Description

This cmdlet creates the specified job type on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
Source	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103.	true	false

JobType	String	This value is the job type name.
		• **Availability for Windows jobs**
		  • **FilesAndFolders**—Files and folders
		  • **ClusterAwareFilesAndFolders**—Cluster-aware files and folders
		  • **SQL**—SQL
		  • **ClusterAwareSql**—Cluster-aware SQL
		  • **FullServerFailover**—Full server
		  • **VRA**—Full server to ESX or full server to Hyper-V
		• **Availability for Linux jobs**
		  • **LinuxFullServerFailover**—Full server
		  • **Lvra**—Full server to ESX
		• **Migrate for Windows jobs**
		  • **MoveDataOnlyMigration**—Files and folders migration
		  • **MoveServerMigration**—Full server migration
		  • **VraMove**—Full server to ESX migration or full server to Hyper-V migration
			true	false

Name	Type	Description	Required	Pipeline Input

		• **Migrate for Linux jobs**
		  • **LinuxMoveServerMigration**—Full server migration
		  • **MoveLvra**—Full server to ESX migration
		• **Other jobs**
		  • **Diagnostics**—Throughput Diagnostic Utility		

JobOptions	JobOptions on page 273	Specify the JobOptions returned from the Get-DtRecommendedJobOptions cmdlet. See Get-DtRecommendedJobOptions on page 62.	false	false
Other Servers	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. Specify multiple server objects in an array using the format @($server1, $server2).	false	false
Create Options	Create Options	Specify the create options available in DoubleTake.Jobs.Contract.CreateOptions. Use the Windows PowerShell New-Object cmdlet to create this object.	true	false

Outputs

Guid on page 267

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName FilesAndFolders

$DtWorkload = Get-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid

$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtServerObjectBeta -Source $DtServerObjectAlpha -JobType FilesAndFolders -Workload $DtWorkload

$DtFnFJobGuid = New-DtJob -ServiceHost $DtServerObjectBeta -Source $DtServerObjectAlpha -JobType FilesAndFolders -JobOptions $DtJobOptions.JobOptions

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called

DtServerObjectBeta. The script then creates a workload on the server for a files and folders job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. The workload definition for the workload type and the server is then stored in the DtWorkload variable. The recommended job options for the servers and the workload type are then stored in the variable DtJobOptions. A new files and folders job is created using the servers and the job options. The job ID is stored in the variable DtFnFJobGuid. The connections for the server object are then closed.

New-DtServer

Creates a server object

Syntax

New-DtServer [-Name] <String> [[-UserName] <String>] [[-Password] <String>] [-Role <String>] [<CommonParameters>]

New-DtServer [-Name] <String> -Credential <PSCredential> [-Role <String>] [<CommonParameters>]

Detailed Description

This cmdlet creates a server object with specific credentials associated with it. This may be any type of server in your organization, for example a Carbonite server, a DNS server, an application server, and so on. This object is used to communicate with the Double-Take Management Service. You should close the connections to this server object when you are finished using it by using Disconnect-DtServer on page 28.

Parameters

Name	Type	Description	Required	Pipeline Input
Name	String	Specify the name or IP address of the server, cluster, or cluster node.	true	false
UserName	String	Specify a user name. If you are using a domain, specify domain\user name.	true	false
Password	String	Specify the password associated with the user you have entered. This password will be visible in plain text.	true	false
Role	String	Specify one of the following roles for the server object you are creating. These servers are used when you specify the -OtherServers parameter in other cmdlets. • **TargetVimServer**—This is the ESX server or vCenter that will host the target appliance and the replica virtual machine for protection and failover. If you are using vCenter, specify your vCenter. Only specify an ESX host if you are using ESX standalone. • **ReverseVimServer**—This is the ESX server or vCenter server that will host the reverse target appliance and the reverse replica virtual machine for reverse protection and failover. If you are using vCenter, specify your vCenter. Only specify an ESX host if you are using ESX standalone. • **ReverseHelperRole**—This is the reverse target	false	false

Name	Type	Description	Required	Pipeline Input
		appliance where data will be replicated during reverse protection.		
Credential	PSCredential on page 308	Specify the credential object returned from the Windows PowerShell Get-Credential cmdlet. This password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.	true	false

Outputs

Server on page 320

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The connections for the server object are then closed.

```
$DtCredentialEncrypted = Get-Credential
$DtServerObjectAlpha = New-DtServer -Name alpha -Credential $DtCredential
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

User credentials are stored in a variable called $DtCredential. The script will prompt you to supply the username and password and the credentials will be encrypted. Then the stored credentials are used to create a new server object for the server alpha. It assigns the server object to the variable called DtServerObject. The connections for the server object are then closed.

New-DtTaskParameters

Creates parameter set

Syntax

New-DtTaskParameters [-ScriptPath] <String> [[-Arguments] <String>] [<CommonParameters>]

Detailed Description

Creates a parameter set to be used with the Invoke-DtQueueTask cmdlet. See Invoke-DtQueueTask on page 92.

Parameters

Name	Type	Description	Required	Pipeline Input
ScriptPath	String	Specify the full path and script name	true	false
Arguments	String	Specify any arguments that need to be passed to the script.	false	false

Outputs

TaskParameters on page 339

Examples

```
$DtScript = New-DtTaskParameters -ScriptPath "C:\PathDir\ScriptName" -Arguments "arg1 arg2"
```

The script called ScriptName, located in C:\PathDir, along with two arguments, is stored in the variable DtScript.

```
$DtPsScript = New-DtTaskParameters -ScriptPath
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -Arguments "-File
""C:\PathDir\Script.ps1"" ""-Arg1 argument1_info -Arg2 argument2_info"" -ExecutionPolicy RemoteSigned"
```

The script to launch PowerShell and run the script called Script.ps1, located in C:\PathDir, along with two arguments and the ExecutionPolicy parameter, is stored in the variable DtPsScript.

New-DtUri

Creates a URI

Syntax

New-DtUri [-Literal] <String> [<CommonParameters>]

New-DtUri [-NetworkId] <String> [-Credential <PSCredential>] [-Port <Int32>] [-Scheme <String>] [-Query <String>] [-Fragment <String>] [<CommonParameters>]

Detailed Description

This cmdlet creates a URI (uniform resource identifier) that is used to specify job credentials

Parameters

Name	Type	Description	Required	Pipeline Input
Literal	String	Specify the entire URI string.	true	false
NetworkId	String	Specify the name or IP address of the server.	true	false
Credential	PSCredential on page 308	Specify the credential object returned from the Windows PowerShell Get-Credential cmdlet. This password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.	false	false
Port	Int32	Specify the communications port.	false	false
Scheme	String	Specify the scheme name.	false	false
Query	String	Specify any additional identification information.	false	false
Fragment	String	Specify any identifying information that provides direction to a secondary resource.	false	false

Outputs

Uri

Examples

```
New-DtUri -Literal "http://server:6320"
```

A URI is created for http://server:6320.

```
New-DtUri -Literal
"foo://username:password@domain.com:6320/location/index?type=volume&directory=C#location "
```

A URI is created for
foo://username:password@domain.com:6320/location/index?type=volume&directory=C#location.

New-DtUvraServer

Creates a server object

Syntax

New-DtUvraServer [-Name] <String> [[-UserName] <String>] [[-Password] <String>] [-Port <Int32>] [-Role <String>] [<CommonParameters>]

New-DtUvraServer [-Name] <String> -Credential <PSCredential> [-Port <Int32>] [-Role <String>] [<CommonParameters>]

Detailed Description

This cmdlet creates a server object with specific credentials associated with it. This cmdlet is specific to the full server to ESX appliance job type, The server object may be one of the following types of servers: Carbonite server, virtual recovery appliance, or VMware host. The object is used to communicate with the Double-Take Management Service.

Parameters

Name	Type	Description	Required	Pipeline Input
Name	String	Specify the name or IP address of the server, cluster, or cluster node.	true	false
Username	String	Specify a user name. If you are using a domain, specify domain\user name.	true	false
Password	String	Specify the password associated with the user you have entered. This password will be visible in plain text.	true	false
Port	Int32	Specify the port for the XML web service protocol. By default, that is 443. Use 6325 for Carbonite servers and appliances, unless you changed the default Carbonite port. Do not specify a port for VMware hosts.	false	false
Role	String	Specify one of the following roles for the server object you are creating. These servers are used when you specify the -OtherServers parameter in other cmdlets. • **TargetVimServer**—This is the ESX server or vCenter that will host the target appliance and the replica virtual machine for protection and failover. If you are using vCenter, specify your vCenter. Only specify an ESX host if you are using ESX standalone. • **ReverseVimServer**—This is the ESX server or vCenter server that will host the reverse target	false	false

Name	Type	Description	Required	Pipeline Input
		appliance and the reverse replica virtual machine for reverse protection and failover. If you are using vCenter, specify your vCenter. Only specify an ESX host if you are using ESX standalone. • **ReverseHelperRole**—This is the reverse target appliance where data will be replicated during reverse protection.		
Credential	PSCredential on page 308	Specify the credential object returned from the Windows PowerShell Get-Credential cmdlet. This password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.	true	false

Outputs

Server on page 320

Examples

```
$DtServerObjectAlpha= New-DtUvraServer -Name alpha -UserName domain\administrator -Password password -Port 6325

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using port 6325 and the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObject. The connections for the server object are then closed.

```
$DtCredentialEncrypted = Get-Credential

$DtServerObjectAlpha = New-DtUvraServer -Name alpha -Credential $DtCredential -Port 6325

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

User credentials are stored in a variable called $DtCredential. The script will prompt you to supply the username and password and the credentials will be encrypted. Then the stored credentials are used to create a new server object for the server alpha using port 6325. It assigns the server object to the variable called DtServerObject. The connections for the server object are then closed.

New-DtWorkload

Creates a workload

Syntax

New-DtWorkload [-ServiceHost] <Server> -WorkloadTypeName <String> [-ImageId <Guid>] [-SnapshotId <Guid>] [<CommonParameters>]

New-DtWorkload [-ServiceHost] <Server> -Workload <Workload> [-ImageId <Guide>] [-SnapshotId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet creates a Carbonite workload on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your source server.	true	false
Workload TypeName	String	Specify a supported workload type from the Get-DtWorkloadType cmdlet. See Get-DtWorkloadType on page 84.	true	false
Workload	Workload on page 365	Specify the workload object returned from the Get-DtWorkload cmdlet. See Get-DtWorkload on page 82.	true	false
ImageId	Guid	This property is no longer used.	false	false
Snapshot Id	Guid	Specify the snapshot GUID returned from the Get-DtSnapshot cmdlet. See Get-DtSnapshot on page 71.	false	false

Outputs

Guid on page 267

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName FilesAndFolders

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The script then creates a workload on the server for a files and folders job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. The connections for the server object are then closed.

Remove-DtJob

Deletes the job

Syntax

Remove-DtJob [-ServiceHost] <Server> [-JobId] <Guid> [[-DeleteOptions] <DeleteOptions>] [<CommonParameters>]

Remove-DtJob [-ServiceHost] <Server> [[-DeleteOptions] <DeleteOptions>] -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet deletes the specified job from the specified server. A running job will be stopped before it is deleted.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Delete Options	Delete Options	Specify the delete options available in DoubleTake.Jobs.Contract.DeleteOptions. Use the Windows PowerShell New-Object cmdlet to create this object.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}
```

```
$DtDeleteOptions = New-Object DoubleTake.Jobs.Contract.DeleteOptions

$DtDeleteOptions.DiscardTargetQueue = $true

Remove-DtJob -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -DeleteOptions
$DtDeleteOptions

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The delete options are stored in DtDeleteOptions, then the specific delete option DiscardTargetQueue is set to true. Finally the job is removed using the delete options. The connections for the server object are then closed.

Remove-DtPhysicalRule

Removes a physical rule

Syntax

Remove-DtPhysicalRule [-ServiceHost] <Server> [-WorkloadId] <Guid> [-Rule] <PhysicalRule> [<CommonParameters>]

Detailed Description

This cmdlet removes a physical rule from the specified workload on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your source server.	true	false
WorkloadId	Guid on page 267	Specify the workload GUID returned from the New-DtWorkload cmdlet using the workload type name parameter. See New-DtWorkload on page 110.	true	false
Rule	PhysicalRule on page 301	Use the Windows PowerShell New-Object cmdlet to create a physical rule object from DoubleTake.Common.Contract.PhysicalRule.	true	false

Outputs

ChangedItems on page 222

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName FilesAndFolders

$DtPhysicalPath = New-Object DoubleTake.Common.Contract.PhysicalRule -Property @ {Path="C:\DirName"}

Remove-DtPhysicalRule -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid -Rule $DtPhysicalPath

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The script then creates a workload on the server for a files and folders job, returning a global unique ID for the workload, and assigns that ID to the variable

DtWorkloadGuid. A new object is created from Double-Take.Common.Contract.PhysicalRule to store the physical path C:\DirName in the variable DtPhysicalPath. Finally, the physical rule is removed from the workload on the server. The connections for the server object are then closed.

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName FilesAndFolders

$DtWorkloadInfo=Get-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid

$DtRemoveRule = $DtWorkloadInfo.PhysicalRules | Where-Object {$_.Path -eq "C:\DirNameToRemove"}

Remove-DtPhysicalRule -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid -Rule $DtRemoveRule

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The script then creates a workload on the server for a files and folders job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. The workload information for the workload is then stored in DtWorkloadInfo. The physical rule within DtWorkloadInfo called C:\DirNameToRemove is then stored in DtRemoveRule. Finally, the physical rule DtRemoveRule is removed from the workload on the server. The connections for the server object are then closed.

Remove-DtSnapshot

Removes a snapshot

Syntax

Remove-DtSnapshot [-ServiceHost] <Server> [-JobId] <Guid> [-SnapshotId] <Guid> [-ConnectionId <Guid>] [<CommonParameters>]

Remove-DtSnapshot [-ServiceHost] <Server> [-JobId] <Guid> [-Snapshot] <SnapshotEntry> [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet removes a Carbonite snapshot from the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
SnapshotId	Guid	Specify the snapshot GUID returned from the Get-DtSnapshot cmdlet. See Get-DtSnapshot on page 71.	true	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
Snapshot	SnapshotEntry on page 329	Specify the snapshot entry object returned from the Get-DtSnapshot cmdlet. See Get-DtSnapshot on page 71.	true	false

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password
```

```
$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtSnaps = Get-DtSnapshot -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

$FirstSnap=$DtSnaps | Select-Object -First 1

Remove-DtSnapshot -ServiceHost $DtServerObjectBeta -SnapshotId $FirstSnap.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The snapshots available for the job are stored in DtSnaps and then the first snapshot is stored in FirstSnap. That first snapshot is then deleted. The connections for the server object are then closed.

Remove-DtSourceQueueSnapshot

Deletes a coordinated snapshot

Syntax

Remove-DtSourceQueueSnapshot [-ServiceHost] <Server> [-CorrelationId] <Guid> [-JobIds] <IEnumerable> [<CommonParameters>]

Detailed Description

This cmdlet deletes a coordinated snapshot for the specified jobs. Snapshots having the same correlation ID but not in the specified jobs will not be deleted.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
CorrelationId	Guid	Specify the correlation ID used to create the coordinated snapshot. See Checkpoint-DtConnectionSourceQueue on page 22.	true	false
JobIds	IEnumerable	Specify an array of job GUIDs. A job GUID is returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false

Outputs

None

Examples

```
$CorId=New-Guid
$Source1Uri = "dtms://112.42.7.63:6325/"
$Source2Uri = "dtms://112.42.7.71:6325/"
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password
$DtJob1 = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object { $_.SourceHostUri -eq $Source1Uri}
$DtJob2 = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object { $_.SourceHostUri -eq $Source2Uri}
$DtJobsArray = New-Object "System.Collections.Generic.List[Guid]"
```

```
$DtJobsArray.Clear()

$DtJobsArray.Add($DtJob1.Id)

$DtJobsArray.Add($DtJob2.Id)

Checkpoint-DtConnectionSourceQueue -ServiceHost $DtServerObjectBeta -CorrelationId $CorId -JobIds
$DtJobsArray.ToArray()

$DtJobsArray2 = New-Object "System.Collections.Generic.List[Guid]"

$DtJobsArray.Clear()

$DtJobsArray2.Add($DtJob2.Id)

Remove-DtSourceQueueSnapshot -ServiceHost $DtServerObjectBeta -CorrelationId $CorId -JobIds
$DtJobsArray2.ToArray()

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

Several variables are set including a unique correlation ID for the snapshots that will be taken as well as URI identifiers for the source servers of existing jobs. A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The jobs are retrieved from DtServerObjectBeta, and the job information is inserted into unique variables. An array of job IDs is then created. A coordinated snapshot is taken using the specified correlation ID and the jobs stored in the array. Only those jobs will have coordinated snapshots. A second array of job IDs is then created, with just one item in the array. The coordinated snapshot for the job in the second array is then deleted. Other snapshots having the same correlation ID but not in the specified array will not be deleted. The connections for the server object are then closed.

Repair-DtJobOptions

Fixes job option errors and warnings

Syntax

Repair-DtJobOptions [-ServiceHost] <Server> [-JobId] <Guid> [-Step] <VerificationStep[]> [<CommonParameters>]

Repair-DtJobOptions [-ServiceHost] <Server> [-CreateOptions] <CreateOptions> [-Step] <VerificationStep[]> [<CommonParameters>]

Repair-DtJobOptions [-ServiceHost] <Server> [-Source] <Server> [-JobType] <String> [-JobOptions] <JobOptions> [-Step] <VerificationStep[]> [-OtherServers <Server[]>] [<CommonParameters>]

Detailed Description

This cmdlet attempts to fix job option errors and warnings. For those errors and warnings that Carbonite cannot correct automatically, you will need to modify the job options manually, modify the source or target configuration, or perhaps select a different target. Use the first syntax for existing jobs and the second syntax for new jobs.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Step	VerificationStep on page 346	Specify the verification steps returned by the Get-DtVerificationStatus cmdlet. See Get-DtVerificationStatus on page 81.	true	true
Create Options	Create Options	Specify the create options available in DoubleTake.Jobs.Contract.CreateOptions. Use the Windows PowerShell New-Object cmdlet to create this object.	true	false
Source	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103.	true	false
JobType	String	This value is the job type name.	true	false

Name	Type	Description	Required	Pipeline Input

		• **Availability for Windows jobs**
		  • **FilesAndFolders**—Files and folders
		  • **ClusterAwareFilesAndFolders**—Cluster-aware files and folders
		  • **SQL**—SQL
		  • **ClusterAwareSql**—Cluster-aware SQL
		  • **FullServerFailover**—Full server
		  • **VRA**—Full server to ESX or full server to Hyper-V
		• **Availability for Linux jobs**
		  • **LinuxFullServerFailover**—Full server
		  • **Lvra**—Full server to ESX
		• **Migrate for Windows jobs**
		  • **MoveDataOnlyMigration**—Files and folders migration
		  • **MoveServerMigration**—Full server migration
		  • **VraMove**—Full server to ESX migration or full server to Hyper-V migration
		• **Migrate for Linux jobs**
		  • **LinuxMoveServerMigration**—Full server migration
		  • **MoveLvra**—Full server to ESX migration
		• **Other jobs**
		  • **Diagnostics**—Throughput Diagnostic Utility		

JobOptions	JobOptions on page 273	Specify the JobOptions returned from the Get-DtRecommendedJobOptions cmdlet. See Get-DtRecommendedJobOptions on page 62.	true	false
Other Servers	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. Specify multiple server objects in an array using the format @($server1, $server2).	false	false

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtValidation = Confirm-DtJobOptions -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -
JobOptions $DtJob.Options

$DtStatus = Get-DtVerificationStatus -ServiceHost $DtServerObjectBeta -Token $DtValidation

$DtRepair = Repair-DtJobOptions -ServiceHost $DtTarget -JobId $DtJob.Id -JobOptions $DtJob.Options -
Step $DtStatus.Steps

Edit-DtJob -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -JobOptions $DtRepair.Options

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The job options used by the job are confirmed, and the validation result is stored in DtValidation. The details of the validation are stored in the variable DtStatus. Ideally you should script this with a loop so you can wait until the verification status is complete before continuing with your script. Those items that can automatically be fixed are corrected. If the job options were modified in order to fix an issues, the updated job options are now contained in the variable $DtRepair. The updated job options are then applied to the job. The connections for the server object are then closed.

Request-DtOnlineActivation

Activates the license

Syntax

Request-DtOnlineActivation -Code <String> -ServerName <String> -ServerInformation <String> [-ServiceHost <Server>] [-EmailAddress <String>] [<CommonParameters>]

Detailed Description

This cmdlet activates the Carbonite license over the Internet using the server information returned by the Get-DtOnlineActivationRequest on page 51 cmdlet.

Parameters

Name	Type	Description	Required	Pipeline Input
Code	String	Specify the 24-character, alpha-numeric license key which applies the appropriate Carbonite license to your Carbonite server. Specify multiple keys in an array using the format @(code1, code2). You can also use the code that is returned by the Get-DtOnlineActivationRequest on page 51 cmdlet.	true	true
Server Name	String	Specify the name of the server or use the server name returned by the Get-DtOnlineActivationRequest on page 51 cmdlet.	true	true
Server Information	String	Specify the server information returned by the Get-DtOnlineActivationRequest on page 51 cmdlet.	true	true
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	false	false
Email Address	String	Specify a valid email address.	false	false

Outputs

ActivationInformation on page 210

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$ServerInfo = Get-DtOnlineActivationRequest -ServiceHost $DtServerObjectAlpha

Request-DtOnlineActivation -Code "1234-5678-9012-3456-7890-1234" -ServerName $DtServerObjectAlpha -
```

```
ServerInformation $ServerInfo.ServerInformation

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The server information for the online activation process is returned. It assigns the server information to the variable called ServerInfo. The license is activated online. The connections for the server object are then closed.

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Get-DtOnlineActivationRequest -ServiceHost $DtServerObjectAlpha | Request-DtOnlineActivation | Set-DtActivationCode

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The server information for the online activation process is returned. The output from the Get-DtOnlineActivationRequest cmdlet is piped directly to Request-DtOnlineActivation, which activates the license online, and then that output (from Request-DtOnlineActivation) is piped directory to Set-DtActivationCode to set the activation key on the server.The connections for the server object are then closed.

Request-DtOnlineDeactivation

Deactivates the license

Syntax

Request-DtOnlineDeactivation [-ServiceHost] <Server> [-EmailAddress <String>] [<CommonParameters>]

Detailed Description

This cmdlet deactivates the Carbonite license over the Internet for the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
Email Address	String	Specify a valid email address.	false	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Request-DtOnlineDectivation -ServerName $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The license for that server is deactivated. The connections for the server object are then closed.

Restart-DtReplicationService

Stop and restarts the Double-Take service

Syntax

Restart-DtReplicationService [-ServiceHost] <Server> [<CommonParameters>]

Detailed Description

This cmdlet stops and restarts the Double-Take service on the specified server. This cmdlet does not impact the Double-Take Management Service.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Restart-DtReplicationService -ServiceHost $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the Double-Take service on the server is stopped and restarted. The connections for the server object are then closed.

Resume-DtJob

Resumes a paused job

Syntax

Resume-DtJob [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Resume-DtJob [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet resumes a paused job. All jobs from the same source to the same IP address on the target will be resumed.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Resume-DtJob -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable

DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The job is then resumed. The connections for the server object are then closed.

Resume-DtMirror

Resumes a paused mirror

Syntax

Resume-DtMirror [-ServiceHost] <Server> [-JobId] <Guid> [-ConnectionId <Guid>] [<CommonParameters>]

Resume-DtMirror [-ServiceHost] <Server> -JobInfo <JobInfo> [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet resumes a paused mirror.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Resume-DtMirror -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The mirror for the job is then resumed. The connections for the server object are then closed.

Resume-DtTarget

Resumes Carbonite processing

Syntax

Resume-DtTarget [-ServiceHost] <Server> -All [<CommonParameters>]

Resume-DtTarget [-ServiceHost] <Server> [-JobId] <Guid> [-ConnectionId <Guid>] [<CommonParameters>]

Resume-DtTarget [-ServiceHost] <Server> -JobInfo <JobInfo> [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet resumes Carbonite processing on the target.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
All	Switch Parameter	Execute the cmdlet on all jobs that are present	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password
```

```
Resume-DtTarget -ServiceHost $DtServerObjectBeta -All
Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. Carbonite processing on that server is then resumed. The connections for the server object are then closed.

Save-DtConsoleServerData

Saves Carbonite Replication Console server data

Syntax

Save-DtConsoleServerData [-FilePath] <String> [<CommonParameters>]

Detailed Description

This cmdlet saves the server information from the Carbonite Replication Console for the user that is currently logged into the machine. If you have multiple administrators sharing the same Carbonite Replication Console installation, each administrator will need to be logged in to save their console server data.

Parameters

Name	Type	Description	Required	Pipeline Input
FilePath	String	Specify the path and file name with an .xml fie extension to store the server data from the Carbonite Replication Console.	true	false

Outputs

None

Examples

```
Save-DtConsoleServerData -FilePath "C:\DtAdmin1_ServerData.xml"
```

The Carbonite Replication Console server data for the user currently logged in will be saved at the root of the C: drive to the file DtAdmin1_ServerData.xml.

Save-DtJobDiagnostics

Saves a job diagnostics file

Syntax

Save-DtJobDiagnostics [-ServiceHost] <Server> [-JobId] <Guid[]> [<CommonParameters>]

Save-DtJobDiagnostics [-ServiceHost] <Server> -JobInfo <JobInfo[]> [<CommonParameters>]

Detailed Description

Saves a job diagnostics file to the \Service\Data\Diagnostics directory of your Carbonite installation on the specified server. The file data identifies the servers in the job, the job information, and the job status at the moment the diagnostics file was created.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

None

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Save-DtJobDiagnostics -Servicehost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The diagnostics files is then saved on the server. The connections for the server object are then closed.

Set-DtActivationCode

Sets the Carbonite license key or activation key

Syntax

Set-DtActivationCode [-ServiceHost] <Server> [-Code] <String[]> [-AdditionalCode <String[]>] [-ActivationKey <String>] [<CommonParameters>]

Detailed Description

This cmdlet sets the Carbonite license key or activation key on the specified server. It also returns the Carbonite license key validation information from the Get-DtActivationStatus cmdlet. See Get-DtActivationStatus on page 32.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	true
Code	String	Specify the 24-character, alpha-numeric license key which applies the appropriate Carbonite license to your Carbonite server. Specify multiple keys in an array using the format @(code1, code2).	true	true
Additional Code	String	Specify any additional keys. Specify multiple keys in an array using the format @(code1, code2).	false	false
Activation Key	String	Specify the 24-character, alpha-numeric activation key which activates your Carbonite license.	false	true

Outputs

ActivationStatus on page 211

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Set-DtActivationCode -ServiceHost $DtServerObjectAlpha -Code 1234567890abcdefghij1234

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the license key 1234567890abcdefghij1234 is applied to the server. The server returns the license key information. The connections for the server object are then closed.

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Get-DtOnlineActivationRequest -ServiceHost $DtServerObjectAlpha | Request-DtOnlineActivation | Set-DtActivationCode

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The server information for the online activation process is returned. The output from the Get-DtOnlineActivationRequest cmdlet is piped directly to Request-DtOnlineActivation, which activates the license online, and then that output (from Request-DtOnlineActivation) is piped directory to Set-DtActivationCode to set the activation key on the server.The connections for the server object are then closed.

Set-DtBandwidthLimit

Sets bandwidth limiting

Syntax

Set-DtBandwidthLimit [-ServiceHost] <Server> [-JobId] <Guid> [-BandwidthLimit] <BandwidthLimit> [-ConnectionId <Guid>] [<CommonParameters>]

Set-DtBandwidthLimit [-ServiceHost] <Server> [-BandwidthLimit] <BandwidthLimit> -JobInfo <JobInfo> [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet sets bandwidth limiting for the specified job .

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Bandwidth Limit	BandwidthLimit on page 217	Specify the bandwidth limit configuration from Get-BandwidthLimit. See Get-DtBandwidthLimit on page 34.	true	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

None

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtFixedBandwidth = Get-DtBandwidthLimit -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

$DtFixedBandwidth.Mode = "Fixed"

$DtFixedBandwidth.Limit = 100000

Set-DtBandwidthLimit -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -BandwidthLimit
$DtFixedBandwidth

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The bandwidth limiting configuration is then stored in DtFixedBandwidth. The Mode is then changed to fixed and the Limit is set to 100,000 bytes/second. Finally, the bandwidth settings are applied to the job. The connections for the server object are then closed.

Set-DtEmailNotificationOptions

Sets e-mail notification configuration

Syntax

Set-DtEmailNotificationOptions [-ServiceHost] <Server> [-Options] <EmailNotificationOptions> [<CommonParameters>]

Detailed Description

This cmdlet sets the Carbonite e-mail notification configuration for the specified server

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
Options	EmailNotificationOptions on page 249	Specify the object returned from the Get-DtEmailNotificationOptions cmdlet. See Get-DtEmailNotificationOptions on page 41.	true	false

Outputs

EmailNotificationOptions on page 249

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtEmailOptions = Get-DtEmailNotificationOptions -ServiceHost $DtServerObject

$DtEmailOptions.Enabled = $true

$DtEmailOptions.SmtpServer = "mail.company.com"

Set-DtEmailNotificationOptions -ServiceHost $DtServerObjectAlpha -Options $DtEmailOptions

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the default Carbonite e-mail notification options are stored in the variable DtEmailOptions. Two of the options are then changed. The Enabled option is set to true which turns on the e-mail notification feature. The SMTP server is also configured for mail.company.com. Finally those changes for the email notification options are set on the server. The connections for the server object are then closed.

Set-DtJobCredentials

Updates credentials

Syntax

Set-DtJobCredentials [-ServiceHost] <Server> [-JobId] <Guid> [-Source <PSCredential>] [-Target <PSCredential>] [-OtherServers <Server[]>] [<CommonParameters>]

Set-DtJobCredentials [-ServiceHost] <Server> -JobInfo <JobInfo> [-Source <PSCredential>] [-Target <PSCredential>] [-OtherServers <Server[]>] [<CommonParameters>]

Detailed Description

Updates the credentials for the source, appliance, and VMware host servers used in the specified job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer or New-DtUvraServer cmdlet. See New-DtServer on page 103 or New-DtUvraServer on page 108.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Source	PSCredential on page 308	Specify the credential object returned from the Windows PowerShell Get-Credential cmdlet. This password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.	false	false
Target	PSCredential on page 308	Specify the credential object returned from the Windows PowerShell Get-Credential cmdlet. This password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.	fasle	false
Other Servers	Server on page 320	Specify the server object returned from the New-DtServer or New-DtUvraServer cmdlet. See New-DtServer on page 103 or New-DtUvraServer on page	false	false

Name	Type	Description	Required	Pipeline Input
		108. Specify the server object in an array using the format @($server).		
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

None

Examples

```
$DtCredentials = Get-Credential domain\administrator

$DtServerObjectAlpha = New-DtServer -Name alpha -Credential $DtCredentials

$DtServerObjectBeta = New-DtServer -Name beta -Credential $DtCredentials

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Set-DtJobCredentials -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -Source $DtCredentials -
Target $DtCredentials

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

You will be prompted for credentials for the domain\administrator account and they will be stored in DtCredentials. Then a server object is created for the servers alpha and beta using the stored credentials. The objects are stored in DtServerObjectAlpha and DtServerObjectBeta, respectively. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Finally, the credentials are updated for the job using the stored credentials. The connections for the server object are then closed.

Set-DtLogicalItemSelection

Adds or removes logical items

Syntax

Set-DtLogicalItemSelection [-ServiceHost] <Server> [-WorkloadId] <Guid> [-LogicalPath] <String> [-Unselect] [<CommonParameters>]

Detailed Description

This cmdlet adds or removes a logical items for the specified workload for the specified server. Adding or removing logical items will add or remove physical rules depending on the workload type.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your source server.	true	false
WorkloadId	Guid on page 267	Specify the workload GUID returned from the New-DtWorkload cmdlet using the workload type name parameter. See New-DtWorkload on page 110.	true	false
Logical Path	String	Specify the path of an item returned from Get-DtLogicalItem. See Get-DtLogicalItem on page 48.	true	false
Unselect	Switch Parameter	Specify this option if you want to remove the logical item.	false	false

Outputs

ChangedItems on page 222

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtWorkloadGuid = New-DtWorkload -ServiceHost $DtServerObjectAlpha -WorkloadTypeName FilesAndFolders

$DtLogicalItems = Get-DtLogicalItem -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGUID

Set-DtLogicalItemSelection -ServiceHost $DtServerObjectAlpha -WorkloadId $DtWorkloadGuid -LogicalPath $DtLogicalItem[0].Path

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. The script then creates a workload on the server for

a files and folders job, returning a global unique ID for the workload, and assigns that ID to the variable DtWorkloadGuid. The logical items associated with the workload type and the server are then stored in the variable DtLogicalItems. Finally, the first logical item in DtLogicalItems is added to the workload. The connections for the server object are then closed.

Set-DtOption

Sets server or job options

Syntax

Set-DtOption [-ServiceHost] <Server> [-Setting] <Hashtable>] [<CommonParameters>]

Set-DtOption [-ServiceHost] <Server> [-Name] <String> -IntValue <Int64> [<CommonParameters>]

Set-DtOption [-ServiceHost] <Server> [-Name] <String> -StringValue <String> [<CommonParameters>]

Set-DtOption [-ServiceHost] <Server> [-Name] <String> -MultiStringValue <String[]> [<CommonParameters>]

Detailed Description

This cmdlet sets the value of the Carbonite server or job option for the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
Setting	Hashtable	Specify a hash table using the format @{option1=value1; option2=value2}. See the *Reference Guide* for details on each job and server option.	true	false
Name	String	Specify the name of the job or server option. See the *Reference Guide* for details on each job and server option. .	true	false
IntValue	Int64	Specify an integer value for the server or job option. See the *Reference Guide* for details on each job and server option.	true	false
StringValue	String	Specify a single string (text) value for the server or job option	true	false
MultiString Value	String	Specify multiple string (text) values for the server or job option in an array using the format @(string1, string2).	true	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Set-DtOption -ServiceHost $DtServerObjectAlpha -Setting @{MaxChecksumBlocks=64;
MirrorChunkSize=131072}

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the server settings MaxChecksumBlocks and MirrorChunkSize are set to 64 and 131072, respectively, on the server. The connections for the server object are then closed.

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Set-DtOption -Servicehost $DtServerObjectAlpha -Name MirrorChunkSize -IntValue 64

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then it sets the MirrorChunkSize server setting to 64. The connections for the server object are then closed.

See Viewing and setting job and server options on page 471 for a sample script that gathers and sets several Carbonite job and server options.

Set-DtPathBlocking

Blocks writing on the target

Syntax

Set-DtPathBlocking [-ServiceHost] <Server> [-SourceAddress] <String> [[-Mode] <PathBlockingMode>] [<CommonParameters>]

Detailed Description

This cmdlet blocks writing to the replica source data located on the target, keeping the data from being changed outside of Carbonite processing.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
Source Address	String	Specify the IP address of the source, including the port, for example, 123.123.123.123:6320.	true	false
Mode	Path Blocking Mode	Specify Blocked or Unblocked.	false	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Set-DtPathBlocking -ServiceHost $DtServerObjectAlpha -SourceAddress "112.42.74.29:6320" -Mode Blocked
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then it sets the target paths associated with the replica data from the specified source IP address to blocked. The connections for the server object are then closed.

Set-DtScriptCredentials

Sets credentials

Syntax

Set-DtScriptCredentials [-ServiceHost] <Server> [-Credential] <PSCredential> [<CommonParameters>]

Set-DtScriptCredentials [-ServiceHost] <Server> [-UserName] <String> [-Password] <String> [<CommonParameters>]

Detailed Description

This cmdlet sets the credentials for Carbonite to use when running scripts on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
Credential	PSCredential on page 308	Specify the credential object returned from the Windows PowerShell Get-Credential cmdlet. This password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.	true	false
UserName	String	Specify a user name. If you are using a domain, specify domain\user name.	true	false
Password	String	Specify the password associated with the user you have entered. This password will be visible in plain text.	true	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtCredentials = Get-Credential domain\administrator

Set-DtScriptCredentials -ServiceHost $DtServerObjectAlpha -Credential $DtCredentials

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then you will be prompted for credentials for the domain\administrator account and those credentials will be stored in the variable DtCredentials. Finally, the credentials used for Carbonite scripts on the server will be set to the stored credentials. The connections for the server object are then closed.

Set-DtServerCredential

Changes server credentials

Syntax

Set-DtServerCredential -Input <Server> -Credential <PSCredential> [<CommonParameters>]

Detailed Description

This cmdlet changes the credentials associated with the server object that has already been created using the New-DtServer cmdlet. See New-DtServer on page 103 for more details on creating a server object.

Parameters

Name	Type	Description	Required	Pipeline Input
Input	String or Server on page 320	Specify the name or IP address of the server, cluster, or cluster node. Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103.	true	false
Credential	PSCredential on page 308	Specify the credential object returned from the Windows PowerShell Get-Credential cmdlet. This password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.	true	false

Outputs

Server on page 320

Examples

```
$DtCredentialEncrypted = Get-Credential
Set-DtServerCredential -Input alpha -Credential $DtCredential
```

User credentials are stored in a variable called $DtCredential. The script will prompt you to supply the username and password and the credentials will be encrypted. Then the stored credentials are used to update the current credentials on the server alpha.

Set-DtVmwareCertificatePolicy

Sets the VMware certificate policy

Syntax

Set-DtVmwareCertificatePolicy [-ServiceHost] <Server> [-VmwareServer] <String> [-Policy] <VmwareCertificatePolicy> [<CommonParameters>]

Detailed Description

This cmdlet sets the VMware certificate policy on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
VmwareServer	String	Specify the name of the VMware server (ESX host or vCenter) where the target server is located.	false	false
Policy	VmwareCertificatePolicy on page 427	Specify the VMware security certificate policy you want to apply.	true	false

Outputs

None

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$TestResult = Test-DtVmwareCertificatePolicy -ServiceHost $DtServerObjectBeta -VmwareServer 112.47.15.6

Set-DtVmwareCertificatePolicy -ServiceHost $DtServerObjectBeta -VmwareServer 112.47.15.6 -Policy AllowAll

Install-DtVmwareCertificate -ServiceHost $DtServerObjectBeta -Certificate $TestResult[1]

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The server $DTServerObjectBeta is tested to see if the current policy or any valid certificates will allow a connection to the specified VMware server. In this example, assume the return is false. The VMware certificate policy on the server $DTServerObjectBeta is then set to allow all certificates to be installed. The certificate from the Test-DtVmwareCertificatePolicy false return is then installed on the server $DTServerObjectBeta. The connections for the server object are then closed.

Start-DtJob

Starts a job

Syntax

Start-DtJob [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Start-DtJob [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet starts the specified job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Start-DtJob -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable

DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The job is then started. The connections for the server object are then closed.

Start-DtJobFailback

Starts failback

Syntax

Start-DtJobFailback [-ServiceHost] <Server> [-JobId] <Guid> [-FailbackOptions] <FailbackOptions> [<CommonParameters>]

Start-DtJobFailback [-ServiceHost] <Server> [-FailbackOptions] <FailbackOptions> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet starts failback for the specified job using the specified failback options.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Failback Options	FailbackOptions on page 256	Specify the failback options returned from the Get-DtRecommendedFailbackOptions cmdlet. See Get-DtRecommendedFailbackOptions on page 58.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}
```

```
$DtFailbackOptions = Get-DtRecommendedFailbackOptions -ServiceHost $DtServerObjectBeta -JobId
$DtJobForAlpha.Id

Start-DtJobFailback -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -FailbackOptions
$DtFailbackOptions

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The failback options are stored in DtFailbackOptions. Failback is then started using the failback options. The connections for the server object are then closed.

Start-DtJobFailover

Starts failover

Syntax

Start-DtJobFailover [-ServiceHost] <Server> [-JobId] <Guid> [-FailoverOptions] <FailoverOptions> [<CommonParameters>]

Start-DtJobFailover [-ServiceHost] <Server> [-FailoverOptions] <FailoverOptions> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet starts failover for the specified job using the specified failover options.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Failover Options	FailoverOptions on page 257	Specify the failover options returned from the Get-DtRecommendedFailoverOptions cmdlet. See Get-DtRecommendedFailoverOptions on page 60.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}
```

```
$DtFailoverOptions = Get-DtRecommendedFailoverOptions -ServiceHost $DtServerObjectBeta -JobId
$DtJobForAlpha.Id

$DtFailoverOptions.FailoverOptions.FailoverMode = 1

Start-DtJobFailover -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -FailoverOptions
$DtFailoverOptions.FailoverOptions

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The failover options are stored in DtFailoverOptions, and then the FailoverMode is set to 1 for a test failover. Failover is then started using the failover options. The connections for the server object are then closed.

Start-DtJobRestore

Starts restoration

Syntax

Start-DtJobRestore [-ServiceHost] <Server> [-JobId] <Guid> [-RestoreOptions] <RestoreOptions> [<CommonParameters>]

Start-DtJobRestore [-ServiceHost] <Server> [-RestoreOptions] <RestoreOptions> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet starts the restoration process for the specified job using the specified restoration options.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Restore Options	RestoreOptions on page 316	Specify the restoration options returned from the Get-DtRecommendedRestoreOptions cmdlet. See Get-DtRecommendedRestoreOptions on page 65.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
```

```
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtRestoreOptions = Get-DtRecommendedRestoreOptions -ServiceHost $DtServerObjectBeta -JobId
$DtJobForAlpha.Id -RestoreTarget $DtServerObjectAlpha

$DtRestoreOptions.RestoreOptions.RestoreParameters.ProcessOrphans = $true

Start-DtJobRestore -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -RestoreOptions
$DtRestoreOptions

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The restoration options are stored in DtRestoreOptions, and then the ProcessOrphans option is set to true. Restoration is then started using the restoration options. The connections for the server object are then closed.

Start-DtJobReverse

Starts reverse

Syntax

Start-DtJobReverse [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Start-DtJobReverse [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet starts the reverse process for the specified job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Start-DtJobReverse -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable

DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The job is then reversed. The connections for the server object are then closed.

Start-DtMirror

Starts mirroring

Syntax

Start-DtMirror [-ServiceHost] <Server> [-JobId] <Guid> [-MirrorParameters] <MirrorParameters> [-ConnectionId <Guid>] [<CommonParameters>]

Start-DtMirror [-ServiceHost] <Server> [-MirrorParameters] <MirrorParameters> -JobInfo <JobInfo> [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet starts mirroring on the specified job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Mirror Parameters	MirrorParameters on page 286	Specify the mirror options available in DoubleTake.Core.Contract.Connection.MirrorParameters. Use the Windows PowerShell New-Object cmdlet to create this object.	true	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtMirrorChecksum = New-Object DoubleTake.Core.Contract.Connection.MirrorParameters

$DtMirrorChecksum.ComparisonCriteria = "Checksum"

$DtMirrorChecksum.Options = "Synchronize,CalculateDifferences"

Start-DtMirror -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -MirrorParameters
$DtMirrorChecksum

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The mirror options are stored in DtMirrorChecksum. The ComparisonCriteria value is changed to checksum and the Options are set to Synchronize and CalculateDifferences. Then the mirror is started for the job using the stored mirroring options. The connections for the server object are then closed.

Start-DtOrphansProcessing

Starts orphan processing

Syntax

Start-DtOrphansProcessing [-ServiceHost] <Server> [-JobId] <Guid> [-ConnectionId <Guid>] [<CommonParameters>]

Start-DtOrphansProcessing [-ServiceHost] <Server> -JobInfo <JobInfo> [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet starts orphan files processing on the specified job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}
```

```
Start-DtOrphansProcessing -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id
Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Orphan file processing is then started for the job. The connections for the server object are then closed.

Start-DtReplication

Starts replication

Syntax

Start-DtReplication [-ServiceHost] <Server> [-JobId] <Guid> [-ConnectionId <Guid>] [<CommonParameters>]

Start-DtReplication [-ServiceHost] <Server> -JobInfo <JobInfo> [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet starts replication on the specified job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Start-DtReplication -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id
```

```
Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Replication is then started for the job. The connections for the server object are then closed.

Start-DtVerify

Starts verification

Syntax

Start-DtVerify [-ServiceHost] <Server> [-JobId] <Guid> [-Synchronize] [-Newer] [-Checksum] [-ProcessOrphans] [-ConnectionId <Guid>] [<CommonParameters>]

Start-DtVerify [-ServiceHost] <Server> -JobInfo <JobInfo> [-Synchronize] [-Newer] [-Checksum] [-ProcessOrphans] [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet starts the Carbonite verification process to check that the replica source data on the target is identical to the actual data on the source

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Synchronize	Switch Parameter	Mirrors to the target any protected files that are different on the source. Without this option, the verification process will only verify the data and generate a verification log file, but it does not remirror any files that are different on the source and target.	false	false
Newer	Switch Parameter	If you are mirroring files to the target during the verification process with the synchronize option, this option will only mirror files that are newer on the source than on the target. If you are using a database application, do not use this option unless you know for certain that you need it. With database applications, it is critical that all files, not just some of the file that might be newer, get mirrored.	false	false
Checksum	Switch Parameter	If you are mirroring files to the target during the verification process with the synchronize option, this option will have the verification process perform a block checksum comparison to determine which	false	false

Name	Type	Description	Required	Pipeline Input
		blocks are different.		
Process Oprhans	Switch Parameter	If you are mirroring files to the target during the verification process with the synchronize option, this option will delete orphaned files on the target.	false	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

None

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Start-DtVerify -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -Synchronize -Newer -Checksum
-ProcessOrphans

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Verification is then started for the job. The connections for the server object are then closed.

Stop-DtJob

Stops a job

Syntax

Stop-DtJob [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Stop-DtJob [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet stops the specified job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Stop-DtJob -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable

DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The job is then stopped. The connections for the server object are then closed.

Stop-DtMirror

Stops mirroring

Syntax

Stop-DtMirror [-ServiceHost] <Server> [-JobId] <Guid> [-ConnectionId <Guid>] [<CommonParameters>]

Stop-DtMirror [-ServiceHost] <Server> -JobInfo <JobInfo> [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet stops mirroring on the specified job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Stop-DtMirror -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The mirror is then stopped for the job. The connections for the server object are then closed.

Stop-DtReplication

Stops replication

Syntax

Stop-DtReplication [-ServiceHost] <Server> [-JobId] <Guid> [-ConnectionId <Guid>] [<CommonParameters>]

Stop-DtReplication [-ServiceHost] <Server> -JobInfo <JobInfo> [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet stops replication on the specified job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Stop-DtReplication -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id
```

```
Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Replication is then stopped for the job. The connections for the server object are then closed.

Stop-DtReplicationService

Stops the Double-Take service

Syntax

Stop-DtReplicationService [-ServiceHost] <Server> [<CommonParameters>]

Detailed Description

This cmdlet stops the Double-Take service on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Stop-DtReplicationService -ServiceHost $DtServerObjectAlpha

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the Double-Take service is stopped on the server. The connections for the server object are then closed.

Suspend-DtJob

Pauses a job

Syntax

Suspend-DtJob [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Suspend-DtJob [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet paused a job. All jobs from the same source to the same IP address on the target will be paused.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Suspend-DtJob -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable

DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The job is then suspended. The connections for the server object are then closed.

Suspend-DtMirror

Pauses mirroring

Syntax

Suspend-DtMirror [-ServiceHost] <Server> [-JobId] <Guid> [-ConnectionId <Guid>] [<CommonParameters>]

Suspend-DtMirror [-ServiceHost] <Server> -JobInfo <JobInfo> [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet pauses mirroring.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Suspend-DtMirror -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The mirror for the job is then paused. The connections for the server object are then closed.

Suspend-DtTarget

Pauses Carbonite processing

Syntax

Suspend-DtTarget [-ServiceHost] <Server> -All [<CommonParameters>]

Suspend-DtTarget [-ServiceHost] <Server> [-JobId] <Guid> [-ConnectionId <Guid>] [<CommonParameters>]

Suspend-DtTarget [-ServiceHost] <Server> -JobInfo <JobInfo> [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet pauses Carbonite processing on the target. Incoming Carbonite data from the source will be queued on the target. All active jobs to the target will complete the operations already in progress. Any new operations will be queued on the target until the target is resumed. The data will not be committed until the target is resumed. Pausing the target only pauses Carbonite processing, not the entire server.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
All	Switch Parameter	Execute the cmdlet on all jobs that are present	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Suspend-DtTarget -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Carbonite processing for the job is then paused. The connections for the server object are then closed.

Test-DtActiveDirectoryCredentials

Tests credentials against Active Directory

Syntax

Test-DtActiveDirectoryCredentials [-ServiceHost] <Server> [-Credential] <PSCredential> -ComputerDomain <String> -ComputerName <String> [<CommonParameters>]

Test-DtActiveDirectoryCredentials [-ServiceHost] <Server> [-UserName] <String> [-Password] <String> -ComputerDomain <String> -ComputerName <String> [<CommonParameters>]

Detailed Description

This cmdlet tests if the specified credentials have privileges to update Active Directory on the specified server's domain.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
Credential	PSCredential on page 308	Specify the credential object returned from the Windows PowerShell Get-Credential cmdlet. This password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.	false	false
Computer Domain	String	Specify a domain name.	true	false
Computer Name	String	Specify a computer name.	true	false
UserName	String	Specify a user name. If you are using a domain, specify domain\user name.	true	false
Password	String	Specify the password associated with the user you have entered. This password will be visible in plain text.	true	false

Outputs

Boolean

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtCredentials = Get-Credential domain\administrator

Test-DtActiveDirectoryCredentials -ServiceHost $DtServerObjectAlpha -Credential $DtCredentials -
ComputerDomain "domain" -ComputerName "alpha"

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then you will be prompted for credentials for the domain\administrator account and those credentials will be stored in the variable DtCredentials. Finally, the stored credentials will be tested to see if they can update Active Directory on the specified domain and computer. The connections for the server object are then closed.

Test-DtEmailNotification

Tests e-mail configuration

Syntax

Test-DtEmailNotification [-ServiceHost] <Server> [-Options] <EmailNotificationOptions> [-To] <String> [-Body] <String> [<CommonParameters>]

Detailed Description

Tests the e-mail options configured with Set-DtEmailNotificationOptions by attempting to send an e-mail to the specified recipient. See Set-DtEmailNotificationOptions on page 140

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
Options	EmailNotificationOptions on page 249	Specify the object returned from the Get-DtEmailNotificationOptions cmdlet. See Get-DtEmailNotificationOptions on page 41.	true	false
To	String	Specify the e-mail address that the test Carbonite e-mail message should be sent to. Multiple addresses can be separated by a comma.	true	false
Body	String	Specify the text of the test Carbonite email message.	true	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
$DtEmailOptions = Get-DtEmailNotificationOptions -ServiceHost $DtServerObject
$DtEmailOptions.Enabled = $true
$DtEmailOptions.SmtpServer = "mail.company.com"
Set-DtEmailNotificationOptions -ServiceHost $DtServerObjectAlpha -Options $DtEmailOptions
```

```
Test-DtEmailNotification -ServiceHost $DtServerObjectAlpha -Options $DtEmailOptions -To
"administrator@mail.company.com" -Body "This is a test Carbonite message."

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the default Carbonite e-mail notification options are stored in the variable DtEmailOptions. Two of the options are then changed. The Enabled option is set to true which turns on the e-mail notification feature. The SMTP server is also configured for mail.company.com. Those changes for the email notification options are set on the server. Finally a test message is sent to the administrator@mail.company.com addresses with the specified message text using the configured e-mail notification options. The connections for the server object are then closed.

Test-DtScript

Tests the specified script

Syntax

Test-DtScript [-ServiceHost] <Server> [-Path] <String> [[-Arguments] <String>] [[-InteractionMode] <DesktopInteractionMode>] [<CommonParameters>]

Detailed Description

This cmdlet tests the specified script on the specified server using the credentials from Set-DtScriptCredentials on page 148. If necessary, manually undo any changes that you do not want after testing the script.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
Path	String	Specify the full path and script name	true	false
Arguments	String	Specify any arguments that need to be passed to the script.	false	false
Interaction Mode	DesktopInteractionMode on page 380	Specify if the script processing will be displayed on the screen, by using the value Interact, or if the script will execute silently in the background, by using the value None.	false	false

Outputs

Int32

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Test-DtScript -ServiceHost $DtServerObjectAlpha -Path "C:\PathDir\ScriptName" -Arguments "arg1 arg2" -InteractionMode Interact

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then the script called ScriptName (located in C:\PathDir) is run, using the arguments arg1 and arg2. The script will display on screen. The connections for the server object are then closed.

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

Test-DtScript -ServiceHost $DtServerObjectAlpha -Path
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -Arguments "-File ""C:\my
scripts\myscript.ps1"" ""-Arg1 arg1 -Arg2 arg2"" -ExecutionPolicy RemoteSigned" -InteractionMode None

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then PowerShell is launched and the arguments passed to PowerShell are the PowerShell script myscript.ps1 (located in C:\my scripts) and the arguments ars1 and arg2. The PowerShell execution policy is set to RemoteSigned so the PowerShell script will execute. The script is set to run silently in the background. The connections for the server object are then closed.

Test-DtScriptCredentials

Tests credentials

Syntax

Test-DtScriptCredentials [-ServiceHost] <Server> [-Credential <PSCredential>] [<CommonParameters>]

Test-DtScriptCredentials [-ServiceHost] <Server> [-UserName] <String> [-Password] <String> [<CommonParameters>]

Detailed Description

This cmdlet tests the specified credentials on the specified server to confirm if they have administrative rights

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost could be your source or target server.	true	false
Credential	PSCredential on page 308	Specify the credential object returned from the Windows PowerShell Get-Credential cmdlet. This password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.	false	false
UserName	String	Specify a user name. If you are using a domain, specify domain\user name.	true	false
Password	String	Specify the password associated with the user you have entered. This password will be visible in plain text.	true	false

Outputs

Boolean

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password

$DtCredentials = Get-Credential domain\administrator

Test-DtScriptCredentials -ServiceHost $DtServerObjectAlpha -Credential $DtCredentials

Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then you will be prompted for credentials for the domain\administrator account and those credentials will be stored in the variable DtCredentials. Finally, the credentials will be tested to confirm if they have administrative rights on the server. The connections for the server object are then closed.

Test-DtVmwareCertificatePolicy

Tests the VMware certificate policy

Syntax

Test-DtVmwareCertificatePolicy [-ServiceHost] <Server> [-VmwareServer] <String> [<CommonParameters>]

Detailed Description

This cmdlet tests to see if the current policy or any valid certificates will allow a connection the specified VMware server. A true return indicates a connection can be made. A false return includes an X509Certificate2 object which can be used in the Install-DtVmwareCertificate on page 89 cmdlet. See X509Certificate2 Class at https://msdn.microsoft.com/en-us/library/system.security.cryptography.x509certificates.x509certificate2(v=vs.110).aspx on the Microsoft MSDN web site for details on the object returned with a false test.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
VmwareServer	String	Specify the name of the VMware server (ESX host or vCenter) where the target server is located.	true	false

Outputs

Boolean

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$TestResult = Test-DtVmwareCertificatePolicy -ServiceHost $DtServerObjectBeta -VmwareServer 112.47.15.6

Set-DtVmwareCertificatePolicy -ServiceHost $DtServerObjectBeta -VmwareServer 112.47.15.6 -Policy AllowAll

Install-DtVmwareCertificate -ServiceHost $DtServerObjectBeta -Certificate $TestResult[1]

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The server $DTServerObjectBeta is tested to see if the current policy or any valid certificates will allow a connection to the specified VMware server. In this example, assume the return is false. The VMware certificate policy on the server $DTServerObjectBeta is then set to allow

all certificates to be installed. The certificate from the Test-DtVmwareCertificatePolicy false return is then installed on the server $DTServerObjectBeta. The connections for the server object are then closed.

Undo-DtJobFailover

Starts undo failover

Syntax

Undo-DtJobFailover [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Undo-DtJobFailover [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet starts the undo failover process for the specified job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Undo-DtJobFailover -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable

DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The undo failover process is then started. The connections for the server object are then closed.

Uninstall-DoubleTake

Uninstalls Carbonite

Syntax

Uninstall-DoubleTake [-RemoteServer] <Server> [-AsJob] [<CommonParameters>]

Detailed Description

This cmdlet uninstalls Carbonite on the specified server.

Parameters

Name	Type	Description	Required	Pipeline Input
Remote Server	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103.	true	false
AsJob	Switch Parameter	Specify if you want the uninstallation to occur asynchronously in the background, returning the PowerShell command immediately. You can get the status of each uninstallation using the Windows PowerShell Get-Job command. Without this parameter, each uninstallation specified will be executed synchronously and the current activity of the current uninstallation will be displayed.	false	false

Outputs

None

Examples

```
$DtServerObjectAlpha= New-DtServer -Name alpha -UserName domain\administrator -Password password
Uninstall-Doubletake -RemoteServer $DtServerObjectAlpha
Disconnect-DtServer -ServiceHost $DtServerObjectAlpha
```

A server object is created for the server alpha using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectAlpha. Then Carbonite is uninstalled from the server. The connections for the server object are then closed.

Update-DtShares

Updates shares

Syntax

Update-DtShares [-ServiceHost] <Server> [-JobId] <Guid> [<CommonParameters>]

Update-DtShares [-ServiceHost] <Server> -JobInfo <JobInfo> [<CommonParameters>]

Detailed Description

This cmdlet updates source share information on the target for the specified job.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

ActivityToken on page 213

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

Update-DtShares -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable

DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. Shares are then updated on the target for the job. The connections for the server object are then closed.

Wait-DtConfirmJobOptions

Waits for the job validation process to complete

Syntax

Wait-DtConfirmJobOptions [-ServiceHost] <Server> -Token <ActivityToken> [-Any] [-Timeout <Int32>] [<CommonParameters>]

Detailed Description

This cmdlet waits for the job validation process triggered by Confirm-DtJobOptions on page 25 to complete before processing any additional cmdlets.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
Token	ActivityToken on page 213	Specify the confirm action object returned from the Confirm-DtJobOption cmdlet. See Confirm-DtJobOptions on page 25.	true	false
Any		Only wait for the first validation to finish before continuing	false	false
Timeout	Int32	Specify the length of time, in seconds, to wait for the process to complete. For example, 120 would wait two minutes and then the next cmdlet would processed. If you set the timeout to zero (0), there is no timeout delay and the next cmdlet is immediately processed. If you do not specify a timeout parameter, the timeout will default to forever.	false	false

Outputs

VerificationStep on page 346

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password

$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtValidation = Confirm-DtJobOptions -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -
```

```
JobOptions $DtJob.Options
Wait-DtConfirmJobOptions -ServiceHost $DtServerObjectBeta -Token $DtValidation
Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The job options used by the job are confirmed, and the validation result is stored in DtValidation. The script then waits until the validation is complete before continuing. The connections for the server object are then closed.

Wait-DtMirrorComplete

Waits for the mirroring process to complete

Syntax

Wait-DtMirrorComplete [-ServiceHost] <Server> [-JobId] <Guid> [-PollingInterval <Int32>] [-ConnectionId <Guid>] [<CommonParameters>]

Wait-DtMirrorComplete [-ServiceHost] <Server> -JobInfo <JobInfo> [-PollingInterval <Int32>] [-ConnectionId <Guid>] [<CommonParameters>]

Detailed Description

This cmdlet waits for the mirroring process to complete before processing any additional cmdlets.

Parameters

Name	Type	Description	Required	Pipeline Input
Service Host	Server on page 320	Specify the server object returned from the New-DtServer cmdlet. See New-DtServer on page 103. For this cmdlet, the -ServiceHost should be your target server.	true	false
JobId	Guid on page 267	Specify the job GUID returned from the New-DtJob cmdlet or the Id within the job information returned from the Get-DtJob cmdlet. See New-DtJob on page 100 and Get-DtJob on page 43.	true	false
Polling Interval	Int32	Specify the amount of time, in hh:mm:ss, to wait before checking to see if the mirror has completed.	false	false
Connection Id	ConnectionId on page 227	Specify the connection ID returned from the Get-DtConnectionIds cmdlet. See Get-DtConnectionIds on page 36.	false	false
JobInfo	JobInfo on page 270	Specify the job information returned from the Get-DtJob cmdlet. The job information can be piped from the Get-DtJob cmdlet and used in this cmdlet. See Get-DtJob on page 43.	true	true

Outputs

MirrorState on page 404

Examples

```
$DtServerObjectBeta = New-DtServer -Name beta -UserName domain\administrator -Password password
```

```
$DtJobForAlpha = Get-DtJob -ServiceHost $DtServerObjectBeta | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtMirrorChecksum = New-Object DoubleTake.Core.Contract.Connection.MirrorParameters

$DtMirrorChecksum.ComparisonCriteria = "Checksum"

$DtMirrorChecksum.Options = "Synchronize,CalculateDifferences"

Start-DtMirror -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -MirrorParameters
$DtMirrorChecksum

Wait-DtMirrorComplete -ServiceHost $DtServerObjectBeta -JobId $DtJobForAlpha.Id -PollingInterval
"00:05:00"

Disconnect-DtServer -ServiceHost $DtServerObjectBeta
```

A server object is created for the server beta using the domain\administrator and password credentials. It assigns the server object to the variable called DtServerObjectBeta. The job(s) are retrieved from DtServerObjectBeta, but only the job information where the source machine name is equivalent to the name stored in the variable DtServerObjectAlpha is retrieved. That information is then stored in the variable DtJobForAlpha. The mirror options are stored in DtMirrorChecksum. The ComparisonCriteria value is changed to checksum and the Options are set to Synchronize and CalculateDifferences. Then the mirror is started for the job using the stored mirroring options. The script will wait for mirroring to complete before continuing. The script will check ever five minutes to see if mirroring is complete. The connections for the server object are then closed.

Chapter 3 Classes

The following classes are used in Carbonite.

- VolumeQualificationResults on page 359
- VRAOptions on page 360
- VRAQualificationResults on page 363
- VRAWorkloadCustomizationOptions on page 364
- Workload on page 365
- WorkloadSupportSummary on page 367
- WorkloadType on page 368

ActivationAttribute

Parameter of

ActivationCode on page 208

Properties

Name	Type	Description
Name	String	This value is the name of the license key attribute.
Value	Int64	This value is the numeric value associated with the license key attribute.

ActivationCode

Parameter of

ActivationStatus on page 211

Properties

Name	Type	Description
Attributes	ActivationAttribute [ ] on page 207	This value represents the attributes of the license.
Code	String	This value is a 24 character, alpha-numeric key which applies the appropriate license to the server.
ExpirationDate	DateTime	The value is the license expiration date.
IsEvaluation	Boolean	This value is true if the license key is an evaluation license; otherwise, it is false.
IsExpired	Boolean	This value is true if the license is expired; otherwise, it is false.
IsNodeLocked	Boolean	This value is true if the license key must be activated; otherwise, it is false.
IsValid	Boolean	This value is true if the license key is valid; otherwise, it is false.
LicenseType	LicenseType on page 401	This value represents the type of license.
MajorVersion	Int32	This value is the major version of the release associated with this license.
MinorVersion	Int32	This value is the minor version of the release associated with this license.
ProductCode	Int32	This value is the product code associated with this license.

Name	Type	Description
ProductName	String	This value is the product name associated with this license.
SerialNumber	Int32	This value is the serial number associated with this license.

ActivationInformation

Returned by

Request-DtOnlineActivation on page 123

Properties

Name	Type	Description
ActivationKey	String	This value is the activation key.
Code	String	This value is the activation key.
EmailAddress	String	This value is the email address to register the activation key.
Error	String	This value is the text for any error messages from the activation process.
Quantity	Int32	This value is the quantity associated with the license.
ServiceHost	Server on page 320	This value is a server object returned from the New-DtServer cmdlet. See New-DtServer on page 103.
ServerName	String	This value is the name of the server.
ServerInformation	String	This value is the unique server information used to generate the activation key for this particular server.

ActivationStatus

Returned by

Get-DtActivationStatus on page 32, Set-DtActivationCode on page 136

Properties

Name	Type	Description
AddOnCodes	ActivationCode [ ] on page 208	This value represents the current add-on licenses on this server.This property is no longer used.
Codes	ActivationCode [ ] on page 208	This value represents the current licenses on this server.
IsNodeLocked	Boolean	This value is true if the license key must be activated; otherwise, it is false.
IsValid	Boolean	This value is true if the license key is valid; otherwise, it is false.

ActivityStatusEntry

Returned by

Get-DtJobActionStatus on page 45

Parameter of

RepairStatus on page 313, VerificationStatus on page 345

Properties

Name	Type	Description
Duration	TimeSpan	This value is the duration of the activity.
MessageFormatParameters	String	This value is the message format parameters.
MessageId	String	This value is the message resource ID.
RequesterUserName	String	The requester is the user who initiated the activity.
Status	ActivityCompletionStatus on page 375	This value represents if or how the activity completed.
TimeStamp	DateTimeOffset	This value is the time stamp for the activity.
Token	ActivityToken on page 213	This value is a unique identifier for the request.

ActivityToken

Returned by

Confirm-DtJobOptions on page 25, Edit-DtJob on page 29, Invoke-DtQueueTask on page 92, Remove-DtJob on page 112, Remove-DtSnapshot on page 116, Repair-DtJobOptions on page 120, Resume-DtJob on page 127, Resume-DtMirror on page 129, Resume-DtTarget on page 131, Start-DtJob on page 153, Start-DtJobFailback on page 155, Start-DtJobFailover on page 157, Start-DtJobRestore on page 159, Start-DtJobReverse on page 161, Start-DtMirror on page 163, Start-DtOrphansProcessing on page 165, Start-DtReplication on page 167, Start-DtVerify on page 169, Stop-DtJob on page 171, Stop-DtMirror on page 173, Stop-DtReplication on page 175, Suspend-DtJob on page 178, Suspend-DtMirror on page 180, Suspend-DtTarget on page 182, Undo-DtJobFailover on page 194, Update-DtShares on page 197

Parameter of

ActivityStatusEntry on page 212, Get-DtVerificationStatus on page 81, Wait-DtConfirmJobOptions on page 199

Properties

Name	Type	Description
ActivityNameFormatParameters	String	This value is the activity format parameters.
ActivityNameId	String	This value is the name of the activity.
Id	Guid	This value is the unique ID of the activity.

ApplicationOptions

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
AagInstances	String	This value is the instance name corresponding to the SQL AlwaysOn Availability Group Server. This value should never contain more than one entry.
DnsRecordLocks	DnsRecordLock	Do not use this property. Carbonite uses it internally.
IsAag	Boolean	Do not use this property. Carbonite uses it internally.
MonitoredServiceRepeatCount	Int32	This property is no longer used. See MonitoringOptions on page 291.ServiceMonitoringOptions on page 328.
MonitoredServices	ServiceInformation	This property is no longer used. See MonitorConfiguration on page 287.ScriptMonitorName.
MonitorScript	String	This property is no longer used. See CoreMonitorOptions on page 237.MonitorConfiguration on page 287.ScriptMonitorName.
RestartService	Boolean	This property is no longer used. See MonitoringOptions on page 291.ServiceMonitoringOptions on page 328.
SourceDomain	String	This value is the source server's primary domain.
SourceName	String	This value is the name of the source server.

Name	Type	Description
TargetDomain	String	This value is the target server's primary domain.
TargetName	String	This value is the name of the target server.
TestPostFailoverScript	String	This value is the path and name of the script to run after the target is brought online during a test failover.
TestPostFailoverScriptArguments	String	This value is the arguments for the TestPostFailoverScript to run.
TestPreFailbackScript	String	This value is the path and name of the script to run before failing back after a test failover.
TestPreFailbackScriptArguments	String	This value is the arguments for the TestPreFailbackScript.

BandwidthEntry

Parameter of

BandwidthOptions on page 218

Properties

Name	Type	Description
DaysOfWeek	Weekdays on page 428	This values is the days of the week to which the entry applies.
EndTime	DateTime	This value is the end time of the entry. It should be between 00:00:00 and 23:59:59 (hh:mm:ss). This value is only applicable if the BandwidthOptions on page 218.BandwidthScheduleMode on page 377 is Scheduled.
EntryType	BandwidthEntryType on page 376	This value represents a daytime or overnight schedule.
IsUnlimited	Boolean	This value is true if the entry allows unlimited bandwidth; otherwise, it is false.
Limit	Int64	This value is the bandwidth limit in bytes per second.
Name	String	This value is the name of the entry. It must be unique among all entries in the bandwidth schedule.
StartTime	DateTime	This value is the start time of the entry. It should be between 00:00:00 and 23:59:59 (hh:mm:ss).

BandwidthLimit

Returned by

Get-DtBandwidthLimit on page 34

Parameter of

Set-DtBandwidthLimit on page 138

Properties

Name	Type	Description
Limit	Int64	This value is the bandwidth limit in bytes per second.
Mode	BandwidthScheduleMode on page 377	This value is the bandwidth limiting mode.

BandwidthOptions

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
Entries	BandwidthEntry [ ] on page 216	This value represents the bandwidth schedule. This value is copied to CoreConnectionOptions.ConnectionStartParameters.Schedule.Bandwidth.Entries.
Limit	Int64	This value is the bandwidth limit in bytes per second.
Mode	BandwidthScheduleMode on page 377	This value is the bandwidth limiting mode. This value is copied to CoreConnectionOptions.ConnectionStartParameters.Schedule.Bandwidth.Mode.
Specifications	BandwidthSpecification [ ] on page 221	This value represents a bandwidth specifications as identified by a common name (for example T1) and a numeric speed (for example, 193,000 bytes per second). This value is copied to CoreConnectionOptions.ConnectionStartParameters.Schedule.Bandwidth.Specification.

BandwidthSchedule

Parameter of

ConnectionSchedule on page 228

Properties

Name	Type	Description
Current	BandwidthScheduleEntry [ ] on page 220	This value is the current bandwidth schedule entry being used.
Entries	BandwidthScheduleEntry [ ] on page 220	This value represents the bandwidth schedule. This value is copied from BandwidthOptions.Entries.
Mode	BandwidthScheduleMode on page 377	This value is the bandwidth limiting mode. This value is copied from BandwidthOptions.Mode.
Specifications	BandwidthSpecification [ ] on page 221	This value represents a bandwidth specifications as identified by a common name (for example T1) and a numeric speed (for example, 193,000 bytes per second). This value is copied from BandwidthOptions.Specification.

BandwidthScheduleEntry

Parameter of

BandwidthSchedule on page 219

Properties

Name	Type	Description
DaysOfWeek	Weekdays on page 428	This values is the days of the week to which the entry applies.
IsUnlimited	Boolean	This value is true if the entry allows unlimited bandwidth; otherwise, it is false.
Limit	Int64	This value is the bandwidth limit in bytes per second.
Name	String	This value is the name of the entry. It must be unique among all entries in the bandwidth schedule.
StartTime	DateTime	This value is the start time of the entry. It should be between 00:00:00 and 23:59:59 (hh:mm:ss).

BandwidthSpecification

Parameter of

BandwidthOptions on page 218, BandwidthSchedule on page 219

Properties

Name	Type	Description
Key	String	This value is a common name that represents a bandwidth specifications, for example T1.
Type	BandwidthSpecificationType on page 378	This value represents the bandwidth specification associated with the type of network (LAN or WAN).
Value	Int64	This value is the bandwidth limit in bytes per second.

ChangedItems

Returned by

Add-DtPhysicalRule on page 16, Remove-DtPhysicalRule on page 114, Set-DtLogicalItemSelection on page 143

Parameter of

Add-DtPhysicalRule on page 16, Add-DtUvraPhysicalRule on page 18, Remove-DtPhysicalRule on page 114

Properties

Name	Type	Description
LogicalItems	LogicalItems [ ] on page 279	This value represents the logical items that changed.
LogicalRules	String [ ]	This value is the current logical rules of the workload.
PhysicalItems	PhysicalItem [ ] on page 299	This value represents the physical items that changed.
PhysicalRules	PhysicalRule [ ] on page 301	This value represents the current physical rules of the workload.

CloudOptions

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
EngineJobType	EngineJobType on page 382	This value is the job type according to the replication engine. (These jobs are not the same as JobInfo on page 270.JobType.)
HardLinkLogPath	String	This value is the path for the hard link processing log / report.

ClusterFilesAndFoldersQualifcationResults

Parameter of

JobQualificationResults on page 275

Properties

Name	Type	Description
TargetFileServerQualificationResults	TargetFileServerQualificationResults on page 334	This value represents target configuration information for clustered files and folders jobs.
TargetGroupDiskOffline	Boolean	This value is true if the disk resource in the target cluster group is offline; otherwise, it is false.

ClusterOptions

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
DependOnResources	String	This value is the dependent resources.
InitializeOnTargetNodeMove	Boolean	Do not use this property. Carbonite uses it internally.
OriginalSourceClusterId	String	This value is the unique ID of the source cluster.
SkipSourceResourceShutdownOnLiveFailover	Boolearn	This property is no longer used. See FailoverOptions on page 257.
SourceClusterName	String	This value is the name of the source cluster.
SourceGroupName	String	The value is the name of the source group.
SourceIsCluster	Boolean	This value is true if the source is a cluster; otherwise, it is false.
TargetClusterName	String	This value is the name of the target cluster.
TargetClusterResourceNamePrefix	String	This value is the optional name of the target cluster resource prefix.
TargetClusterStoragePath	String	This value is the location of the target cluster storage.
TargetGroupName	String	This value is the name of the target group.
TargetIsCluster	Boolean	This value is true if the target is a cluster; otherwise, it is false.
TargetVirtualServers	String	This value is the name of the target virtual servers.

CompressionLevel

Parameter of

ConnectionStartParameters on page 229, CoreQualificationResults on page 238, RestoreParameters on page 317

Properties

Name	Type	Description
Algorithm	Int32	The algorithm and level properties are used together in the following combinations. • **Disabled**—Compression is disabled if the level equals -1. The algorithm value is ignored when level equals -1. • **Low compression**—Compression is enabled at a low level if level equals 0 and algorithm equals 10. • **Medium compression**—Compression is enabled at a medium level if level equals 1 and algorithm equals 21. • **High compression**—Compression is enabled at a high level if level equals 2 and algorithm equals 31.
Level	Int32	

ConnectionId

Returned by

Get-DtConnectionIds on page 36

Parameter of

Get-DtBandwidthLimit on page 34, Invoke-DtQueueTask on page 92, Remove-DtSnapshot on page 116, Resume-DtMirror on page 129, Resume-DtTarget on page 131, Set-DtBandwidthLimit on page 138, Start-DtMirror on page 163, Start-DtOrphansProcessing on page 165, Start-DtReplication on page 167, Start-DtVerify on page 169, Stop-DtMirror on page 173, Stop-DtReplication on page 175, Suspend-DtMirror on page 180, Suspend-DtTarget on page 182, Wait-DtMirrorComplete on page 201

Properties

Name	Type	Description
Key	String	This value is the description of the associated connection ID.
Value	Guid	This value is a unique ID of a connection associated with a job.

ConnectionSchedule

Parameter of

ConnectionStartParameters on page 229

Properties

Name	Type	Description
Archive	None	This property is no longer used.
Bandwidth	BandwidthSchedule on page 219	This value represents the bandwidth schedule.
Orphans	OrphansSchedule on page 295	This value represents if orphan files (files in the target path location that are not present on the source) will be deleted. There is currently no schedule (time-related options) for orphan files. They can only be deleted or left on the target.
Transmission	None	This property is no longer used.
Verify	VerifySchedule on page 347	This value represents the verification schedule.

ConnectionStartParameters

Parameter of

CoreConnectionOptions on page 235

Properties

Name	Type	Description
ArchiveBinLocation	String	This property is no longer used.
CompressionLevel	CompressionLevel on page 226	This value represents the compression to use for data passed over the connection.
IsEncrypted	Boolean	This value is true if the connection data is to be encrypted; otherwise, it is false.
IsMirrorEnabled	Boolean	This value is true if a mirror is started when the connection is created; otherwise, it is false.
IsPathBlockingEnabled	Boolean	This value is true if the destination paths on the target are blocked for writing, except by Carbonite; otherwise, it is false.
IsReplicationEnabled	Boolean	This value is true if replication is started when the connection is created; otherwise, it is false.
IsRestore	Boolean	This value is true if the connection is a restoration mirror; otherwise, it is false.
MirrorParameters	MirrorParameters on page 286	This value represents the mirror parameters used for the connection. If this value is null, the default mirror parameters will be used.
Schedule	ConnectionSchedule on page 228	This value represents the bandwidth, orphans, and verification schedule. These schedule settings will be applied to all connections that share the same route to the same target. Therefore, if a connection exists over the same route to the

Name	Type	Description
		same target as the connection you are creating, any schedule changes will be applied to that connection as well.
ScriptPoints	ScriptPoint [ ] on page 319	This value represents the scripts to be run at specific points during mirroring.
SnapshotSchedule	SnapshotSchedule on page 330	This value represents the snapshot schedule for the target.

CoreConnectionDetails

Parameter of

FullServerJobDetails on page 264, JobStatistics on page 276

Properties

Name	Type	Description
BandwidthCollar	Int32	This value is the current bandwidth limit. It may vary according to the bandwidth limit schedule.
CompressionEnabled	Boolean	This value is true if compression is enabled; otherwise, it is false.
CompressionLevel	Int32	This value is the current compression level.
ConnectionId	Int32	This value is the ID assigned to the connection. The ID is a one-based index that resets each time the Double-Take service is restarted.
CurrentTime	DateTimeOffset	This value is the time the connection detail was retrieved.
DiskQueueBytes	Int64	This value is the number of bytes queued on the target.
Encrypted	Boolean	This value is true if Carbonite is encrypting data before sending it to the target; otherwise, it is false.
InitialMirrorComplete	Boolean	This value is true if the initial mirror has completed; otherwise, it is false.
ManagedConnectionId	Guid	This value is the ID assigned to the connection by the Management Service.
MirrorBytesRemaining	Int64	This value is the number of remaining mirror bytes.
MirrorBytesSent	Int64	This value is the number of mirror bytes sent.

Name	Type	Description
MirrorBytesSkipped	Int64	This value is the number of mirror bytes skipped. Bytes are skipped because the data is not different on the source and target.
MirrorBytesTransmitted	Int64	The value is the number of mirror bytes transmitted.
MirrorEndTime	DateTimeOffset	This value is the time when mirroring ended.
MirrorOpsQueued	Int64	This value is the number of mirror operations queued on the target.
MirrorPermillage	Int16	This value is the percentage of the mirror that is complete. A mirror may be at 99-100% when it is actually still processing . For example, this may occur if files were added after the protected data set size was calculated or if there are alternate data streams in the protected data set.
MirrorStartTime	DateTimeOffset	This value is the time when mirroring started.
MirrorState	MirrorState on page 404	This value represents the state of mirroring.
PeerMemoryLow	Boolean	This value is true if memory on the target is low; otherwise it is false.
ReplicationBytesQueued	Int64	This value is the number of replication bytes queued on the target.
ReplicationBytesSent	Int64	This value is the number of replication bytes sent.
ReplicationBytesTransmitted	Int64	The value is the number of replication bytes transmitted.
ReplicationOpsQueued	Int64	This value is the number of replication operations queued on the target.
ReplicationSetCalcInProgress	Boolean	This value is true if the size of the protected data set is being calculated; otherwise, it is false. Mirroring will start while the size of the protected data set is being determined. While the calculation is in progress, the remaining byte count is not valid.

Name	Type	Description
ReplicationState	ReplicationState on page 411	This value represents the state of replication.
Restoring	Boolean	This value is true if the connection is restoring data; otherwise, it is false.
SourceAccessLevel	AccessLevel on page 372	This value is the source security access. This value will be null if the Carbonite access level cannot be determined. If the value is not null and SourceAvailable is false, Carbonite can connect to the source but was not able to determine at least Carbonite monitor access with the provided credentials.
SourceAvailable	Boolean	This value is true if the target can communicate with the source; otherwise, it is false.
SourceClusterResourceState	ClusterResourceState on page 379	This value represents the state of the source cluster resource.
SourceEndpoint	String	This value is the source endpoint that is the route from the target back to the source.
SourceEngineAvailable	Boolean	This value is true if the source replication engine is available; otherwise, it is false.
SourceMachineName	String	This value is the name of the source server.
SourceRecoveryPointLatency	Int64	This value is the length of time replication is behind on the target compared to the source. This is the time period of data that would be lost if a failure were to occur at the current time.
SourceRecoveryPointTime	DateTimeOffset	This value is the time that replication is synchronized between the source and target. The difference between the current time and this time is the time period of data that would be lost if a failure were to occur at the current time.
SourceUniqueId	String	This value is a unique ID assigned to the source.
StartTime	DateTimeOffset	This value is the time the connection was started.

Name	Type	Description
TargetAccessLevel	AccessLevel on page 372	This value is the target security access. This value will be null if the Carbonite access level cannot be determined. If the value is not null and TargetAvailable is false, Carbonite can connect to the target but was not able to determine at least Carbonite monitor access with the provided credentials.
TargetAvailable	Boolean	This value is true if the target is available; otherwise, it is false.
TargetEngineAvailable	Boolean	This value is true if the target replication engine is available; otherwise, it is false.
TargetLoaded	Boolean	This value is true if the Carbonite target module is loaded on the server; otherwise, it is false.
TargetMachineName	String	This value is the name of the target server.
TargetQueueBytes	Int64	This value is the number of bytes in the target queue.
TargetRoute	String	This value is the endpoint on the target that is the route the source is using for the connection.This route is in the format [address]:[port].
TargetState	TargetStates on page 424	This value represents the state of the target.
TargetUniqueId	String	This value is a unique ID assigned to the target.
TotalBytesSent	Int64	This value is the total number of bytes that have been sent for the connection.
TotalBytesTransmitted	Int64	This value is the total number of bytes that have been transmitted for the connection.
TotalOpsQueued	Int64	This value is the total number of operations queued on the target.
TransmissionMode	TransmissionMode on page 426	This value represents the state of data transmission from the source to the target.

CoreConnectionOptions

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
ConnectionStartParameters	ConnectionStartParameters on page 229	This value represents the parameters used to start the connection between the source and target.
PathTransformations	PathTransformation [ ] on page 298	This value represents where the source protected data will be located on the target.
TargetAddress	String	This value is the target IP address.
TargetEnginePort	Int32	This value is the port where the engine is listening.

CoreMonitorDetails

Parameter of

JobStatistics on page 276

Properties

Name	Type	Description
HighAvailabilityState	HighAvailabilityState on page 396	This value represents the state of failover monitoring.
MonitoredAddressStatuses	MonitoredAddressStatus [ ] on page 290	This value represents the status of a monitored IP address.
MonitorId	Guid	This value is a unique ID for this failover monitor.
MonitorName	String	This value is the name for this failover monitor.
RestoreStates	RestoreStates on page 413	This value represents the state of a restoration connection.
TargetAvailable	Boolean	This value is true if the target is available; otherwise, it is false.

CoreMonitorOptions

Parameter of

FailbackOptions on page 256, JobOptions on page 273

Properties

Name	Type	Description
MonitorConfiguration	MonitorConfiguration on page 287	This value represents the configuration of failover monitoring.
ShouldPerformLanFailover	Boolean	This value is for display in the console to determine which failover option should be selected by default.
SourceDomain	String	This value is the name of the source domain.
SourceServer	String	This value is the name of the source server.
TargetDomain	String	This value is the name of the target domain.
TargetServer	String	This value is the name of the target server.
TotalTimeAllowed	TimeSpan	This value is the amount of time allowed without a successful ping before a failover condition is met.
UseTotalTimeAllowed	Boolean	This value is true if TotalTimeAllowed is used to determine when a failover condition is met; otherwise, it is false.

CoreQualificationResults

Parameter of

JobQualificationResults on page 275

Properties

Name	Type	Description
CompressionLevels	CompressionLevel [ ] on page 226	This value is the compression level. It will be null if there is a problem getting the compression level from the source or target.
DefaultAllToOneBasePath	String	This value sets the default base path for the all to one path mapping, which configures the source replica data to be stored on a single volume on the target. The default is \source_ name\volume_name. If you are protecting multiple volumes on the source, each volume would be stored on the same volume on the target. For example, C:\data and D:\files for the source Alpha would be stored in D:\alpha\C and D:\alpha\D, respectively.
SourceEnginePort	Int32	This value is the port where the engine is listening.
SourceIPAddresses	UnicastIPAddressInfo [ ] on page 343	This value represents the available IP addresses on the source.
SourceMachineName	String	This value is the name of the source server.
SourceNetworkId	String	This value is the source network ID used to communicate with the source from the target.
SourceNetworkInterfaces	NetworkInterfaceInfo [ ] on page 292	This value represents the available NICs on the source.
SourceProductInfo	ProductInfo on page 305	This value represents the Carbonite product info on the source.
SourceVolumes	Volume [ ] on page 353	This value represents the available volumes on the source.

Name	Type	Description
TargetEnginePort	Int32	This value is the port where the engine is listening.
TargetIPAddresses	UnicastIPAddressInfo [ ] on page 343	This value represents the available IP addresses on the target. These IP addresses are the possible values that can be used to populate CoreConnectionOptions on page 235.TargetAddress.
TargetMachineName	String	This value is the name of the target server.
TargetNetworkInterfaces	NetworkInterfaceInfo [ ] on page 292	This value represents the available NICs on the target.
TargetProductInfo	ProductInfo on page 305	This value represents the Carbonite product info on the target.
TargetVolumes	Volume [ ] on page 353	This value represents the available volumes on the target.

Credentials

Parameter of

DnsDomainDetails on page 246, EmailNotificationOptions on page 249, MonitorConfiguration on page 287, Server on page 320

Properties

Name	Type	Description
Domain	String	This value is the name of the domain.
Password	String	This value is the password.
UserName	String	This value is the name of the user.

CutoverDetails

Parameter of

FullServerJobDetails on page 264

Properties

Name	Type	Description
PercentComplete	Int32	This value is the percentage of the cutover that is complete.
State	Int32	This value is the state of the cutover. If this value is 1, the cutover is idle. If this value is 1000, the cutover processing is complete. Negative numbers are error codes.

DeleteOptions

Returned by

Get-DtUvraRecommendedRemoveOptions on page 79

Properties

Name	Type	Description
DeleteOnClusterResourceOffline	Boolean	This value is true if the job is deleted because the target cluster resource went offline; otherwise, it is false.
DeleteReplica	Boolean	This value is true if the replica virtual machine should be deleted when the job is deleted; otherwise, it is false.
DiscardTargetQueue	Boolean	This value is true if any operations in the target queue should be deleted when the job is deleted; otherwise, it is false.
ImageOptions	None	This property is no longer used.

Disk

Parameter of

ServerInfo on page 323

Properties

Name	Type	Description
DiskName	String	This value is the name of the disk.
Partitions	Partition [ ] on page 296	This value is the partitions on the disk.
PartTableType	String	This value is the partition table type.
SizeInB	Int64	This value is the size, in bytes, of the disk.
Valid	Boolean	This value is true if the disk is valid; otherwise, it is false.

DiskOptions

Parameter of

TestFailoverOptions on page 340, VRAOptions on page 360

Properties

Name	Type	Description
DesiredSizeInB	String	This value is the desired size, in bytes, of the new disk.
DiskControllerType	String	This value is the disk controller. • **IDE**—Use this value to create an IDE disk. • **SCSI**—Use this value to create a SCSI disk.
DiskName	String	This value is the name of the disk.
DiskProvisioningType	String	This value is the disk type. • **Dynamic**—Use this value for ESX thin disks and for Hyper-V dynamic disks. • **Fixed**—Use this value for ESX thick disks and for Hyper-V fixed disks. • **Flat Disk**—Use this value for ESX flat disks. This type is not supported on Hyper-V.
Partitions	Partition [ ] on page 296	This value is the partitions on the disk.
PartTableType	String	This value is the partition table type.
PreexistingDiskPath	String	This value is the full path and file name of an existing virtual disk that you want to reuse. If this value is null, a new virtual disk will be created.
SizeInB	Int64	This value is the size, in bytes, of the disk.

Name	Type	Description
Valid	Boolean	This value is true if the disk is valid; otherwise, it is false.
VirtualDiskPath	String	This value is the path on the host where the disk should be stored.

DnsDomainDetails

Parameter of

DnsOptions on page 247

Properties

Name	Type	Description
Credentials	Credentials on page 240	This value represents the domain credentials.
DnsServers	DnsServerDetail [ ] on page 248	This value represents a DNS server that will be updated during DNS failover.
DomainName	String	This value is the name of the domain.
IpAddressMappings	IpAddressMap [ ] on page 268	This value represents the mapping between source IP address and target IP address when using DNS failover.
ShouldUpdateTtl	Boolean	This value is true if the time to live value should be updated; otherwise, it is false.
TtlValue	Int32	This value is the time to live value in seconds.

DnsOptions

Returned by

Get-DtDnsOptions on page 38

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
AdditionalDns	String	This value identifies any additional source DNS servers to update.
AllIpAddressesNeedMapping	Boolean	This value is true if all IP addresses need to be mapped to a target IP address; otherwise, it is false.
AlternateTrustee	String	This value is the source server trustee, which overrides the default trustees of "NT AUTHORITY\SYSTEM" and the source computer account (for clustered environments).
Domains	DnsDomainDetails [ ] on page 246	This value represents the domains that will be updated during failover.
Enabled	Boolean	This value is true if DNS failover is enabled; otherwise, it is false.
SourceServerInWorkgroup	Boolean	This value is true if the source is in a workgroup; otherwise, it is false.
SourceServerName	String	This value is the name of the source server.
TargetServerName	String	This value is the name of the target server.

DnsServerDetail

Parameter of

DnsDomainDetails on page 246

Properties

Name	Type	Description
Address	String	This value is the IP address of the DNS server.
Origin	String	This value is the origin of the DNS server, which indicates if the DNS server was discovered from the source server, the target server, both servers, or manually specified by the user.
SelectedForUpdate	Boolean	This value is true if the DNS server will be updated during failover; otherwise, it is false.

EmailNotificationOptions

Returned by

Get-DtEmailNotificationOptions on page 41, Set-DtEmailNotificationOptions on page 140

Parameter of

Test-DtEmailNotification on page 186

Properties

Name	Type	Description
ConnectionSecurity	SmtpConnectionSecurity on page 418	This value represents the SMTP server connection security.
Enabled	Boolean	This value is true if email notification is enabled; otherwise, it is false.
EntryIdFilter	String	This value is one or more comma-delimited Windows Event Log entry IDs for which email notifications should be sent.
EntryTypeFilter	String	This value is one or more comma-delimited Windows Event Log entry types (error, warning, or information) for which email notifications should be sent.
From	String	This value is the e-mail address that will be placed in the From field of the email message.
IncludeEventDescriptionInSubject	Boolean	This value is true if the event description is included in the email subject; otherwise, it is false.
LoginToSmtpServer	Boolean	This value is true if you must log in to the SMTP server; otherwise, it is false.
SmtpCredentials	Credentials on page 240	This value represents the credentials used to log into the SMTP server. Do not set this value if you want to user the SMTP server anonymously.

Name	Type	Description
SmtpPort	Int32	This value is the SMTP port. The default port is 25.
SmtpServer	String	This value is the name of the SMTP server.
SubjectPrefix	String	This value is the text that will be added to the beginning of the email subject.
To	String	This value can contain one or more comma-delimited e-mail addresses to which the email notifications will be sent.

EngineControlStatus

Parameter of

JobStatus on page 277

Properties

Name	Type	Description
CanPauseMirror	Boolean	This value is true if mirroring can be paused;otherwise, it is false.
CanPauseTarget	Boolean	This value is true if the target can be paused;otherwise, it is false.
CanPauseTransmission	Boolean	This value is true if transmission can be paused;otherwise, it is false.
CanProcessOrphans	Boolean	This value is true if the orphan file removal process can be started;otherwise, it is false.
CanResumeMirror	Boolean	This value is true if mirroring can be resumed;otherwise, it is false.
CanResumeTarget	Boolean	This value is true if the target can be resumed;otherwise, it is false.
CanResumeTransmission	Boolean	This value is true if transmission can be resumed;otherwise, it is false.
CanSetBandwidth	Boolean	This value is true if a bandwidth limit can be set;otherwise, it is false.
CanStartMirror	Boolean	This value is true if mirroring can be started;otherwise, it is false.
CanStartTransmission	Boolean	This value is true if transmission can be started;otherwise, it is

Name	Type	Description
		false.
CanStopMirror	Boolean	This value is true if mirroring can be stopped;otherwise, it is false.
CanTakeSnapshot	Boolean	This value is true if the target can take a snapshot of the replica data;otherwise, it is false.
CanUpdateShares	Boolean	This value is true if source share information can be updated;otherwise, it is false.
CanVerify	Boolean	This value is true if verification of the source data to the target replica data can be started;otherwise, it is false.
ConnectionId	Guid	This value is a unique ID that needs to be provided to the replication engine control methods for controlling the replication engine connection.
Role	String	This value is the role of the replication engine for the job. Jobs may use more than one replication engine connection in their operation. This property provides a way to differentiate between the replication engine connections for display purposes.

EventLogEntry

Returned by

Get-DtEventLogEntry on page 42

Properties

Name	Type	Description
Category	String	This value is the text associated with the CategoryNumber.
CategoryNumber	Int16	This value is the category number of the event.
EntryType	EventLogEntryType on page 254	This value is the type of event.
Index	Int32	This value is the index of this event in the event log.
InstanceId	Int32	The value is a resource identifier that designates the message text for the event.
MachineName	String	This value is the name of the server that generated the event.
Message	String	This value is the localized event message.
ReplacementStrings	String	This value includes any replacement strings associated with the event.
Source	String	This value is the name of the application that generated the event.
TimeGenerated	DateTime	This value is the local time when the event was generated.
TimeWritten	DateTime	This value is the local time when the event was written to the event log.
UserName	String	This value is the name of the user that generated the event.

EventLogEntryType

Parameter of

EventLogEntry on page 253

Properties

Name	Type	Description
Error	String	The event message is an error message.
Information	String	The event message is an information message.
Warning	String	The event message is a warning message.

ExtendedLowLevelStates

Parameter of

JobStatus on page 277

Properties

Name	Type	Description
Health	Health on page 395	This value represents the health of the low level state. Possible uses include displaying an icon for a low level state.
MessageId	String	This value is a unique ID for the state message.
MessageIdFormatParameters	String	This value is the state message.

FailbackOptions

Parameter of

RecommendedFailbackOptions on page 309, Start-DtJobFailback on page 155

Properties

Name	Type	Description
NewMonitorOptions	CoreMonitorOptions on page 237	This value represents the options for configuring failover monitoring after a job has been restored to a new source server.

FailoverOptions

Returned by

Get-DtUvraRecommendedFailoverOptions on page 77

Parameter of

RecommendedFailoverOptions on page 310, Start-DtJobFailover on page 157

Properties

Name	Type	Description
FailoverDataAction	FailoverDataAction on page 384	This value represents the action to take on the queued target data before failover.
FailoverMode	FailoverMode on page 387	This value represents the live, test, or snapshot style of failover.
FailoverType	FailoverType.Options on page 393	This value represents the automatic or manual style of failover.
PerformTestFailover	Boolean	This option is no longer used. Use FailoverMode on page 387 instead.
SkipSourceResourceShutdown	Boolean	This value is true if the cluster resource is not shut down on the source during live failover; otherwise, it is false.
SnapshotId	Guid	This value is the snapshot ID that should be applied to the target during snapshot failover.

FailoverReport

Returned by

Get-DtAllFailoverReports on page 33, Get-DtLatestFailoverReport on page 47

Properties

Name	Type	Description
failoverSuccess	Boolean	This value is true if the failover was successful; otherwise, it is false.
failoverType	FailoverStyle on page 390	This value represents a live or test failover.
isReverse	Boolean	This value is true if this is the reverse of a job; otherwise, it is false.
jobId	String	This value is the unique ID for the job.
jobName	String	This value is the name of the job.

jobType	String	This value is the job type name.
		• **Availability for Windows jobs**
		  • **FilesAndFolders**—Files and folders
		  • **ClusterAwareFilesAndFolders**—Cluster-aware files and folders
		  • **SQL**—SQL
		  • **ClusterAwareSql**—Cluster-aware SQL
		  • **FullServerFailover**—Full server
		  • **VRA**—Full server to ESX or full server to Hyper-V
		• **Availability for Linux jobs**
		  • **LinuxFullServerFailover**—Full server
		  • **Lvra**—Full server to ESX
		• **Migrate for Windows jobs**

Name	Type	Description
		• **MoveDataOnlyMigration**—Files and folders migration • **MoveServerMigration**—Full server migration • **VraMove**—Full server to ESX migration or full server to Hyper-V migration • **Migrate for Linux jobs** • **LinuxMoveServerMigration**—Full server migration • **MoveLvra**—Full server to ESX migration • **Other jobs** • **Diagnostics**—Throughput Diagnostic Utility
snapshotLabel	String	This value is the label of the snapshot being used. If you are performing a live failover, this value will be null.
targetedMachineInfo	MachineInfoClass on page 285	This value represents the target server's operating system and version.
times	TimeClass on page 342	This value represents the time at various points during failover.

FailoverScriptConfiguration

Parameter of

MonitorConfiguration on page 287

Properties

Name	Type	Description
PostFailbackScript	String	This value is the full path and file name of the script located on the target and to be run on the target after failback.
PostFailbackScriptArgs	String	This value is the comma-separated argumentsfor PostFailbackScript.
PostFailoverScript	String	This value is the full path and file name of the script located on the target and to be run on the target after failover.
PostFailoverScriptArgs	String	This value is the comma-separated argumentsfor PostFailoverScript.
PreFailbackScript	String	This value is the full path and file name of the script located on the target and to be run on the target before failback.
PreFailbackScriptArgs	String	This value is the comma-separated argumentsfor PreFailbackScript.
PreFailbackWait	Boolean	This value is true if Carbonite will wait for PreFailbackScript to complete before continuing with the failback operation; otherwise, it is false.
PreFailoverScript	String	This value is the full path and file name of the script located on the target and to be run on the target before failover.
PreFailoverScriptArgs	String	This value is the comma-separated argumentsfor PreFailoverScript.

Name	Type	Description
PreFailoverWait	Boolean	This value is true if Carbonite will wait for PreFailoverScript to complete before continuing with the failover operation; otherwise, it is false.
SourcePostFailbackScript	String	This value is the full path and file name of the script located on the source and to be run on the source after failback.
SourcePostFailbackScriptArgs	String	This value is the comma-separated argumentsfor SourcePostFailbackScript.

Feature

Parameter of

JobInfo on page 270

Properties

Name	Type	Description
Endpoint	Uri	This value is the endpoint that identifies the location of the service.
Tag	String	This value is the tag that identifies the role of the service.

FullServerFailoverOptions

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
AdditionalStagingFolders	String	This value is the folders that should be staged on the target. They will be applied to their actual installation location during failover.
CreateBackupConnection	Boolean	This value is true if reverse protection should be configured for this job; otherwise, it is false.
RemoveOrphans	Boolean	This value is true if orphan files should be removed for the reverse connection; otherwise, it is false.
ReverseRoute	String	This value is the source IP address to be used for the reverse connection.
ReverseRouteEnginePort	Int32	This value is the port where the engine is listening.
ShutdownSourceServer	Boolean	This value is true if the source server should be shut down during cutover; otherwise, it is false.
SourceChecksumAll	Boolean	This value is true if a checksum should be used for all files when comparing them; otherwise, it is false.

FullServerJobDetails

Parameter of

JobStatistics on page 276

Properties

Name	Type	Description
BackupConnectionDetails	CoreConnectionDetails on page 231	This value represents the details of the Carbonite Availability reverse connection.
CutoverDetails	CutoverDetails on page 241	This value represents the Carbonite Migrate cutover detail.
ProtectionConnectionDetails	CoreConnectionDetails on page 231	This value represents the details of the protection connection.
RecoveryConnectionDetails	None	This property is no longer used.
SystemVolumeRevertConnectionDetails	CoreConnectionDetails on page 231	This value represents the details of the system volume revert connection.

FullServerNicMappings

Parameter of

FullServerTestFailoverOptions on page 266, SystemStateOptions on page 332

Properties

Name	Type	Description
SourceNic	String	This value is the NIC on the source server.
TargetNic	String	This value is the NIC on the target server.
TargetNicList	String [ ]	This value is the list of NICs on the target server.
TestFailoverTargetNic	String	This value is the NIC on the test failover server.
TestFailoverTargetNicList	String [ ]	This value is the list of NICs on the test failover server.

FullServerTestFailoverOptions

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
DeleteSnapshots	Boolean	This value is true if snapshots will be deleted after the test failover; otherwise, it is false.
TestFailoverServerCredential	TestFailoverServerCredentials on page 341	This value represents the server to use for test failover for full server jobs.
NicMappings	FullServerNicMappings [ ] on page 265	This value represents the NIC mappings for the test failover for the full server job.
FailoverName	String	This value is the name applied to the test failover server. If the job settings are failing over the server name, the source name will be used on the test server. If not, the target name will be retained on the test server.
TestFailoverServerAddress	String	This value is the IP address used for the test failover connection.
TestFailoverServerEnginePort	Int32	This value is the port used on the test failover server.
TestFailoverServerReservedAddress	String	This value is the reserved IP address on the test server

Guid

Returned by

New-DtFilesAndFoldersJob on page 98, New-DtJob on page 100, New-DtWorkload on page 110

Parameter of

Add-DtPhysicalRule on page 16, Checkpoint-DtConnection on page 20, Close-DtWorkload on page 24, Confirm-DtJobOptions on page 25, Edit-DtJob on page 29, Get-DtBandwidthLimit on page 34, Get-DtConnectionIds on page 36, Get-DtJob on page 43, Get-DtJobActionStatus on page 45, Get-DtLogicalItem on page 48, Get-DtQualificationResults on page 56, Get-DtRecommendedFailbackOptions on page 58, Get-DtRecommendedFailoverOptions on page 60, Get-DtRecommendedPathTransform on page 64, Get-DtRecommendedRestoreOptions on page 65, Get-DtSnapshot on page 71, Get-DtUvraRecommendedFailoverOptions on page 77, Get-DtUvraRecommendedRemoveOptions on page 79, Get-DtWorkload on page 82, Get-DtWorkloadPhysicalItem on page 83, Invoke-DtQueueTask on page 92, Remove-DtJob on page 112, Remove-DtPhysicalRule on page 114, Remove-DtSnapshot on page 116, Repair-DtJobOptions on page 120, Resume-DtJob on page 127, Resume-DtMirror on page 129, Resume-DtTarget on page 131, Save-DtJobDiagnostics on page 134, Set-DtBandwidthLimit on page 138, Set-DtJobCredentials on page 141, Set-DtLogicalItemSelection on page 143, Start-DtJob on page 153, Start-DtJobFailback on page 155, Start-DtJobFailover on page 157, Start-DtJobRestore on page 159, Start-DtJobReverse on page 161, Start-DtMirror on page 163, Start-DtOrphansProcessing on page 165, Start-DtReplication on page 167, Start-DtVerify on page 169, Stop-DtJob on page 171, Stop-DtMirror on page 173, Stop-DtReplication on page 175, Suspend-DtJob on page 178, Suspend-DtMirror on page 180, Suspend-DtTarget on page 182, Undo-DtJobFailover on page 194, Update-DtShares on page 197, Wait-DtMirrorComplete on page 201

Properties

Name	Type	Description
Guid	String	This value is a unique ID.

IpAddressMap

Parameter of

DnsDomainDetails on page 246

Properties

Name	Type	Description
ShouldUpdateTtl	Boolean	This value is true if the time to live value should be updated; otherwise, it is false.
SourceIP	String	This value is the IP address on the source server
TargetIP	String	This value is the IP address on the target server
TtlValue	Int32	This value is the time to live value in seconds.

JobAction

Parameter of

JobStatus on page 277

Properties

Name	Type	Description
Duration	TimeSpan	This value is the duration of the action.
ErrorCode	Int32	This value is the error code associated with the action.
ExceptionMessage	String	This value is a message from the underlying exception.The message stored in this property should not be displayed in the user interface, as it may not be available in the language used by the interface. This property should only be used for internal troubleshooting, and as a measure of last resort in the event of a completely unexpected error.
Id	Guid	This value is the unique ID of the action.
MessageFormatParameters	String	This value is the message format parameters.
MessageId	String	This value is the message ID for the action. This value can be used to look up a message associated with this action.
RequestingUserName	String	This value is the name of the user that requested the action.
Status	ActionStatus on page 373	This value represents the status of the action.
Timestamp	DateTimeOffset	This value is the time when the action occurred.
TitleFormatParameters	String	This value is the title format parameters for the action.
TitleId	String	This value is the title ID for the action. This value can be used to look up a title associated with this action.

JobInfo

Returned by

Get-DtJob on page 43

Parameter of

Checkpoint-DtConnection on page 20, Confirm-DtJobOptions on page 25, Edit-DtJob on page 29, Get-DtBandwidthLimit on page 34, Get-DtJob on page 43, Get-DtQualificationResults on page 56, Get-DtRecommendedFailbackOptions on page 58, Get-DtRecommendedFailoverOptions on page 60, Get-DtRecommendedRestoreOptions on page 65, Get-DtSnapshot on page 71, Get-DtUvraRecommendedFailoverOptions on page 77, Get-DtUvraRecommendedRemoveOptions on page 79, Invoke-DtQueueTask on page 92, Remove-DtJob on page 112, Remove-DtSnapshot on page 116, Resume-DtJob on page 127, Resume-DtMirror on page 129, Resume-DtTarget on page 131, Save-DtJobDiagnostics on page 134, Set-DtBandwidthLimit on page 138, Set-DtJobCredentials on page 141, Start-DtJob on page 153, Start-DtJobFailback on page 155, Start-DtJobFailover on page 157, Start-DtJobRestore on page 159, Start-DtJobReverse on page 161, Start-DtMirror on page 163, Start-DtOrphansProcessing on page 165, Start-DtReplication on page 167, Start-DtVerify on page 169, Stop-DtJob on page 171, Stop-DtMirror on page 173, Stop-DtReplication on page 175, Suspend-DtJob on page 178, Suspend-DtMirror on page 180, Suspend-DtTarget on page 182, Undo-DtJobFailover on page 194, Update-DtShares on page 197, Wait-DtMirrorComplete on page 201

Properties

Name	Type	Description
CreatorUserName	String	This value is the name of the user who created the job.
Features	Feature [ ] on page 262	This value represents a service endpoint that provides additional features for a job.
Id	Guid	This value is the unique ID for the job.
JobPersistedState	None	Do not use this property. Carbonite uses it internally.

JobType	String	This value is the job type name.
		• **Availability for Windows jobs**
		  • **FilesAndFolders**—Files and folders
		  • **ClusterAwareFilesAndFolders**—Cluster-aware files and folders

Name	Type	Description
		• **SQL**—SQL • **ClusterAwareSql**—Cluster-aware SQL • **FullServerFailover**—Full server • **VRA**—Full server to ESX or full server to Hyper-V • **Availability for Linux jobs** • **LinuxFullServerFailover**—Full server • **Lvra**—Full server to ESX • **Migrate for Windows jobs** • **MoveDataOnlyMigration**—Files and folders migration • **MoveServerMigration**—Full server migration • **VraMove**—Full server to ESX migration or full server to Hyper-V migration • **Migrate for Linux jobs** • **LinuxMoveServerMigration**—Full server migration • **MoveLvra**—Full server to ESX migration • **Other jobs** • **Diagnostics**—Throughput Diagnostic Utility
LoadedFromDisk	Boolean	This value is true if the job was loaded from persistence; otherwise, it is false.
Managed	Boolean	This value is true if the job is managed; otherwise, it is false. This property is intended to be used by transitional implementations of jobs that want to allow their jobs to be visible to the job manager, but will be responsible for creating those jobs and managing their lifetime.
Options	JobOptions on page 273	This value represents the current options for the job.

Name	Type	Description
OtherHostUris	IDictionary	This value is the URIs for other servers involved in the job.
SourceHostUri	URI	This value is the source URI.
SourceUniqueId	String	This value is the unique ID of the source server.
Statistics	JobStatistics on page 276	This value represents the current statistics of the job.
Status	JobStatus on page 277	This value represents the current state of the job.
TargetHostUri	Uri	This value is the target URI.
TargetUniqueId	String	This value is the unique ID of the target server.

JobOptions

Parameter of

Confirm-DtJobOptions on page 25, Edit-DtJob on page 29, JobInfo on page 270, New-DtFilesAndFoldersJob on page 98, New-DtJob on page 100, RecommendedJobOptions on page 311, RepairStatus on page 313

Properties

Name	Type	Description
ApplicationOptions	ApplicationOptions on page 214	This value represents the options for SQL jobs.
BandwidthOptions	BandwidthOptions on page 218	This value represents bandwidth limiting configuration.
CloudOptions	CloudOptions on page 223	The values represents cloud job options.
ClusterOptions	ClusterOptions on page 225	This value represents cluster options.
CoreConnectionOptions	CoreConnectionOptions on page 235	This value represents the options for the job connection.
CoreMonitorOptions	CoreMonitorOptions on page 237	This value represents the options for configuring failover monitoring by using the service monitoring method.
DnsOptions	DnsOptions on page 247	This value represents the options for jobs using DNS failover.
DtavOptions	None	This property is no longer used.
DTHVOptions	None	This property is no longer used.
FailoverMonitoringEnabled	Boolean	This value is true if failover monitoring is enabled; otherwise, it is false.
FullServerFailoverOptions	FullServerFailoverOptions on page 263	This value represents the options for full server jobs.
FullServerTestFailoverOptions	FullServerTestFailoverOptions on page 266	This values represents the test failover options for full server jobs.

Name	Type	Description
ImageProtectionOptions	None	This property is no longer used.
ImageRecoveryOptions	None	This property is no longer used.
ManagementServiceTargetPort	Int32	This value is the port where the engine is listening.
MonitoringOptions	MonitoringOptions on page 291	This value represents the options for configuring failover monitoring by using the service monitoring method.
Name	String	This value is the name of the job.
SimpleFailoverMonitorOptions	None	This property is no longer used.
SourceNetworkIdForManagementService	String	This property is the IP address or host name that will be used for Double-Take Management Service communication from the target to the source.
SystemStateOptions	SystemStateOptions on page 332	This value represents the system state options for full server jobs.
TargetServicesOptions	TargetServicesOptions on page 335	This value represents which target services will be stopped and started during failover and failback.
UnmanagedConnectionOptions	UnmanagedConnectionOptions on page 344	This value represents the options for unmanaged connections which are connections that are not associated with a known job. These connections were created in a client outside of the job manager.
VcdVappOptions	None	This property is no longer used.
VRAOptions	VRAOptions on page 360	This value represents the options for guest-level virtual jobs.
Workload	Workload on page 365	This value represents the workload being protected by the job.

The JobOptions class includes all possible configurations for all possible job types, which makes it large and potentially confusing. See Carbonite Replication Console Set Options page to JobOptions class mapping on page 477 to help identify which JobOptions classes correspond to job options in the Carbonite Replication Console.

JobQualificationResults

Returned by

Get-DtQualificationResults on page 56

Parameter of

RecommendedJobOptions on page 311

Properties

Name	Type	Description
ClusterFilesAndFoldersQualificationResults	ClusterFilesAndFoldersQualifcationResults on page 224	This value represents configuration information for clustered files and folders jobs.
CoreQualificationResults	CoreQualificationResults on page 238	This value represents the replication engine connection configuration.
DtavQualificationResults	None	This property is no longer used.
DTHVQualificationResults	None	This property is no longer used.
SourceBehindNat	Boolean	This value is true if the source is located behind a NAT router; otherwise, it is false.
SourceHostUri	Uri	This value is the source URI.
SupportsFrameworkMonitoring	Boolean	This value is true if the server supports failover monitoring; otherwise, it is false.
SuppressFailoverMonitorOptions	Boolean	This value is true if failover monitoring options are suppressed; otherwise, it is false.
VcdVappQualificationResults	None	This property is not applicable. It is reserved for future features.
VRAQualificationResults	VRAQualificationResults on page 363	This value represents configuration information for guest-level virtual jobs.

JobStatistics

Parameter of

JobInfo on page 270

Properties

Name	Type	Description
CoreConnectionDetails	CoreConnectionDetails on page 231	This value represents the details of the connection.
CoreMonitorDetails	CoreMonitorDetails on page 236	This value represents the details of failover monitoring.
FullServerJobDetails	FullServerJobDetails on page 264	This value represents the details of a full server job.
ImageProtectionJobDetails	None	This property is no longer used.
ImageRecoveryJobDetails	None	This property is no longer used.

JobStatus

Parameter of

JobInfo on page 270

Properties

Name	Type	Description
Actions	JobAction [ ] on page 269	This value represents an action recently reported by a job.
CanCreateImageRecovery	None	This property is no longer used.
CanDelete	Boolean	This value is true if you can delete the job; otherwise, it is false.
CanEdit	Boolean	This value is true if the job can be edited; otherwise, it is false.
CanFailback	Boolean	This value is true if the job can be failed back; otherwise, it is false.
CanFailover	Boolean	This value is true if the job can be failed over; otherwise, it is false.
CanPause	Boolean	This value is true if the job can be can be paused; otherwise, it is false.
CanRestore	Boolean	This value is true if the job can be restored; otherwise, it is false.
CanReverse	Boolean	This value is true if the job can be reversed; otherwise, it is false.
CanStart	Boolean	This value is true if the job can be started; otherwise, it is false.
CanStop	Boolean	This value is true if the job can be stopped; otherwise, it is false.

Name	Type	Description
CanUndoFailover	Boolean	This value is true if the failover of the job can be undone; otherwise, it is false.
EngineControlStatuses	EngineControlStatus on page 251	This value represents the status of the replication engine.
ExtendedLowLevelState	ExtendedLowLevelStates on page 255	This value represents additional information about the low level job state.
Health	Health on page 395	This value represents the high level health of the job.
HighLevelState	HighLevelState on page 397	This value represents high level job states. High level states typically map to phases of the job's overall lifecycle, like mirroring or failing over. Jobs will pass through various high level states during their lifetime, and the types of high level states are usually shared between jobs, regardless of job type.
IsInError	Boolean	This value is true if the job is in an error state; otherwise, it is false.
LowLevelState	String	This value is a low level job state. Low level states typically map to individual steps the job is performing at any given time, like attaching virtual disks or updating DNS. Low level states provide detailed information about a job's current state or action and will vary between different job types.
PermillageComplete	Int32	This value is the percentage of the current activity for that job that is complete. To display a percentage string for an arbitrary activity, jobs can provide a value for this property, and that value will be formatted and displayed in the client. To remove the display of the percentage, set this property to zero.
TargetState	String	This value is derived from TargetStates on page 424. Information from the connection is distilled into a string key which is sent to the client for display.

LogicalItems

Returned by

Get-DtLogicalItem on page 48

Parameter of

ChangedItems on page 222, Get-DtLogicalItem on page 48

Properties

Name	Type	Description
IsContainer	Boolean	This value is true if the item is a container of other items; otherwise, it is false.
IsReadOnly	Boolean	This value is true if the item is read-only (meaning the Saturation cannot be changed); otherwise, it is false.
ItemType	String	This value is the type of item.
Metadata	String	This value is any additional metadata that may be displayed for a particular type of item. This metadata is not in any particular format, but you must be able to determine the format and decipher the metadata based on the type.
Name	String	This value is the name of the item. Typically, this name is used for display in the user interface.
PartitionName	String	This value is the name of the partition.
Path	String	This value is the path of the item.
Saturation	SaturationLevel on page 415	This value represents the saturation level of the item. Typically, this value is used to display a visual state in the user interface.
SaturationDefault	SaturationLevel on page 415	This value is the default saturation level of the item. Typically, this value is used to display a visual state in the user interface.

LogicalVolume

Parameter of

VolumeGroup on page 355

Properties

Name	Type	Description
Attributes	FileSystemAttributes on page 394	This value represents the file system attributes.
AvailableFreeSpace	Int64	This value is the amount of free space on the volume.
CreationTime	DateTime	This value is the time when the volume was created.
DesiredSize	Int64	This value is the desired size of the new volume.
DiskControllerType	String	This value is the disk controller. • **IDE**—Use this value to create an IDE disk. • **SCSI**—Use this value to create a SCSI disk.
DiskProvisoningType	String	This value is the disk type. • **Dynamic**—Use this value for ESX thin disks and for Hyper-V dynamic disks. • **Fixed**—Use this value for ESX thick disks and for Hyper-V fixed disks. • **Flat Disk**—Use this value for ESX flat disks. This type is not supported on Hyper-V.
DriveFormat	String	This value is the file system format of the volume.
DriveType	DriveType	This value is the drive type.
IsContainer	Boolean	This value is true if the volume is a container of other volumes; otherwise, it is false.

Name	Type	Description
IsReadOnly	Boolean	This value is true if the volume is read-only; otherwise, it is false.
IsSupported	Boolean	This value is true if the volume type is supported; otherwise, it is false.
IsSystemDrive	Boolean	This value is true if the volume is the system volume; otherwise, it is false.
ItemType	String	This value is the type of item.
Label	String	This value is the label assigned to the volume.
LastAccessTime	DateTime	This value is the time when the volume was last accessed.
LastWriteTime	DateTime	This value is the time when the volume was last written to.
LogicalVolumeName	String	This value is the name of the logical volume.
Metadata	String	This value is any additional metadata that may be displayed for a particular type of item. This metadata is not in any particular format, but you must be able to determine the format and decipher the metadata based on the type.
Name	String	This value is the name of the volume. Typically, this name is used for display in the user interface.
Path	String	This value is the path of the volume.
PreexistingDiskPath	String	This value is the full path and file name of an existing virtual disk that you want to reuse. If this value is null, a new virtual disk will be created.
Saturation	SaturationLevel on page 415	This value represents the saturation level of the volume. Typically, this value is used to display a visual state in the user interface.

Name	Type	Description
ShortNameBehavior	Boolean	Do not use this property. Carbonite uses it internally.
Size	Int64	This value is the physical size of the volume.
TotalSize	Int64	This value is the total size of the volume.
VirtualDiskPath	String	This value is the path on the host where the volume should be stored.
VolumeSignature	Int16	This value is the volume signature.
VolumeType	String	This value is the type of volume.

LogMessage

Returned by

Get-DtLogMessage on page 49

Properties

Name	Type	Description
Hash	Int32	This value is the message hash which is used to differentiate messages written with the same Timestamp.
Id	Int32	This value is the message ID.
MessageType	String	This value is the type of log message.
ProcessId	Int32	This value is the process ID of the process that generated the message.
Sequence	Int32	This value is the sequence number of this message in the log file.
Source	String	This value identifies the service (Double-Take or Double-Take Management Service) that generated the log message.
Text	String	This value is the content of the log message.
ThreadId	Int32	This value is the thread ID of the thread that generated the message.
Timestamp	DateTime	This value is the local time when the message was generated.
TimeStampOffset	DateTimeOffset	This value is the relative UTC time when the message was generated.

LvmOptions

Parameter of

ServerInfo on page 323, TestFailoverOptions on page 340, VRAOptions on page 360

Properties

Name	Type	Description
VolumeGroup	VolumeGroup [ ] on page 355	This value represents the volume group properties.

MachineInfoClass

Parameter of

FailoverReport on page 258

Properties

Name	Type	Description
os	String	This value is the server's operating system.
osVersion	String	This value is the operating system version.

MirrorParameters

Parameter of

ConnectionStartParameters on page 229, Start-DtMirror on page 163

Properties

Name	Type	Description
ComparisonCriteria	MirrorComparisonCriteria on page 402	This value represents how to compare the source protected data and the replica on the target.
Options	MirrorOperationOptions on page 403	This value represents what mirror operation will be performed.
OverrideJobOrphansProcessing	Boolean	This value is true if the job's orphan file setting is to be overridden; otherwise, it is false.

MonitorConfiguration

Parameter of

CoreMonitorOptions on page 237

Properties

Name	Type	Description
ActiveDirectoryCredentials	Credentials on page 240	This value represents the Active Directory credentials.
ActiveDirectoryOptions	ActiveDirectoryFailoverOptions on page 374	This value represents the Active Directory options.
Addresses	MonitoredAddressConfiguration [ ] on page 289	This value represents the configuration of the IP addresses that should be monitored for failure.
DataAction	FailoverDataAction on page 384	This value represents the action to take on the queued target data before failover.
FailoverIPAddressOption	FailoverIPAddressesOption on page 385	This value represents which IP addresses should be applied to the target server during failover.
FailoverType	FailoverType.Monitor on page 392	This value represents if an entire server is being failed over.
ItemsToFailover	FailoverItems on page 386	This value represents what will be failed over to the target.
MaxScriptFailures	Int32	This value is the number of times a monitoring script can fail when using MonitoredAddressConfiguration on page 289.PingMethods on page 408.Script.
Name	String	This value is the name of the failover monitor.
ProcessingOptions	FailoverProcessingOptions on page 388	This value represents how the failover is processed.
ReplaceActions	FailoverReplaceActions on page 389	This value represents what identity items are to be replaced on the target during failover.

Name	Type	Description
ScriptMonitorEngine	String	This property is no longer used.
ScriptMonitorName	String	This value is the full path and name of a monitoring script when using MonitoredAddressConfiguration on page 289.PingMethods on page 408.Script.
Scripts	FailoverScriptConfiguration on page 260	This value represents the scripts that should be run at different stages of failover and failback.
SourceEndpoint	String	This value is the IP endpoint used to communicate with Carbonite on the source.
SSMLogPath	String	This value is the path of the system state (full server processor) log.
SSMManualReboot	Boolean	This value is true if the server has to be manually rebooted to apply the system state after failover; otherwise, it is false.
SSMRecoveryType	None	This property is no longer used.
SSMSourceNicGuids	String	This value is the global unique IDs associated with the source NICs.
SSMStagingPath	String	This value is the path of the staged folders on the target.
SSMTargetNicGuids	String	This value is the global unique IDs associated with the target NICs.
Trigger	FailoverTrigger on page 391	This value represents when a failover operation is triggered with respect to the monitored IP addresses.

MonitoredAddressConfiguration

Parameter of

MonitorConfiguration on page 287

Properties

Name	Type	Description
Address	String	This value is the monitored IP address.
EnginePort	Int32	This value is the port where the engine is listening.
MacAddress	String	This value is the MAC address of the NIC for the monitored IP address.
MaxPingAttempts	Int16	This value is the maximum number of ping attempts that will be attempted before considering the IP address to be failed.
NicName	String	This value is the name of the NIC. On a Windows server, this is typically a global unique ID.
PingInterval	TimeSpan	This value is the interval for how often the monitored IP address is pinged.
PingMethods	PingMethods on page 408	This value represents how the IP address is pinged.
SubnetMask	String	This value is the subnet mask of an IPv4 IP address.

MonitoredAddressStatus

Parameter of

CoreMonitorDetails on page 236

Properties

Name	Type	Description
Address	String	This value is the monitored IP address.
Alive	Boolean	This value is true if the IP address is responding to pings; otherwise, it is false.
FailoverConditionMet	Boolean	This value is true if the IP address is considered failed; otherwise, it is false.
RemainingTime	TimeSpan	This value is the amount of time remaining before the IP address is considered failed.
WarningConditionMet	Boolean	This value is true if the IP address is in a warning condition; otherwise, it is false. An IP address is considered to be in a warning condition if it has failed to respond to ping attempts for three quarters of the number of attempts allowed to be missed before the IP address is considered failed. For example, if the value for MonitoredAddressConfiguration on page 289.MaxPingAttempts is 20, then this property will be true after 15 failed attempts.

MonitoringOptions

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
ServiceMonitoringEnabled	Boolean	This value is true if the service monitoring method for failover monitoring is enabled; otherwise it is false.
ServiceMonitoringOptions	ServiceMonitoringOptions on page 328	This value represents the options used for the service monitoring method for failover monitoring.

NetworkInterfaceInfo

Parameter of

CoreQualificationResults on page 238, ServerInfo on page 323

Properties

Name	Type	Description
Description	String	This value is the description of the NIC.
DnsDomain	String	This value is the DNS domain assigned to the NIC.
DnsServers	String	This value is the DNS server assigned to the NIC.
Gateways	String	This value is the gateway assigned to the NIC.
Guid	String	This value is the unique ID for the NIC. It is equivalent to the SettingID property of the Win32_NetworkAdapterConfiguration WMI class as well as the NetworkInterface.Id property in the Base Class Library (BCL).
Index	Int32	Do not use this property. Carbonite uses it internally.
InterfaceIndex	Int32	Do not use this property. Carbonite uses it internally.
IPAddresses	UnicastIPAddressInfo [ ] on page 343	This value represents the IP addresses assigned to the NIC.
MacAddresses	String	This value represents the MAC address of the NIC.
Name	String	This value is the friendly name of the NIC.
PnpInstanceId	String	This value is the plug and play instance ID of the NIC.
ServiceName	String	This value is the service name of the NIC.

OperatingSystemInfo

Parameter of

ServerInfo on page 323

Properties

Name	Type	Description
Architecture	OperatingSystemArchitecture on page 405	This value represents the operating system architecture.
CSDVersion	Int32	This value is the version of the operating system service pack as a number.
HasBCDTemplate	Boolean	This value is true if the BCD template file exists; otherwise it is false.
ProductSuite	Int32	This value is the Windows OSVERSIONINFOEX structure, which contains operating system version information including major and minor version numbers, a build number, a platform identifier, and information about product suites and the latest service pack installed on the system.
ProductType	OperatingSystemProductType on page 406	This value represents the type of operating system.
ServicePack	String	This value is the version of the operating system service pack as a string.
Version	OperatingSystemVersion on page 294	This value is the version of the operating system as a number.
VersionString	String	This value is the version of the operating system as a string.

OperatingSystemVersion

Parameter of

OperatingSystemInfo on page 293

Properties

Name	Type	Description
Build	Int32	This value is the build number of the operating system.
Major	Int32	This value is the major version of the operating system.
Minor	Int32	This value is the minor version of the operating system.
Revision	Int32	This value is the revision number of the operating system.

OrphansSchedule

Parameter of

ConnectionSchedule on page 228

Properties

Name	Type	Description
IsEnabled	Boolean	This value is true if orphan files (files in the target path location that are not present on the source) will be deleted; otherwise, it is false. There is currently no schedule (time-related options) for orphan files. They can only be on or off.

Partition

Parameter of

Disk on page 243

Properties

Name	Type	Description
FileSystem	String	This value is the file system for the partition.
Flags	String	This value is the flags for the partition.
PartitionEndInB	Int64	This value is the point in bytes where the partition ends.
PartitionName	String	This value is the name of the partition.
PartitionNumber	Int32	This value is the partition number.
PartitionStartInB	Int64	This value is the point in bytes where the partition begins.
PartitionType	String	This value represents the partition type like primary, extended, or logical.
Valid	Boolean	This value is true if the partition is valid; otherwise, it is false.

PathBlocking

Returned by

Get-DtPathBlocking on page 53

Properties

Name	Type	Description
BlockingMode	PathBlockingMode on page 407	This value represents if the path is blocked.
Paths	String [ ]	This value is the paths that are blocked.
SourceAddress	String	This value is the source IP address that corresponds to the connection that has the blocked paths.

PathTransformation

Returned by

Get-DtRecommendedPathTransform on page 64

Parameter of

CoreConnectionOptions on page 235, ReplicaVmInfo on page 314, RestoreParameters on page 317, VmInfo on page 352

Properties

Name	Type	Description
SourcePath	String	This value is the path on the source.
TargetPath	String	This value is the path on the target.

PhysicalItem

Returned by

Get-DtPhysicalItem on page 54, Get-DtWorkloadPhysicalItem on page 83

Parameter of

ChangedItems on page 222, Get-DtPhysicalItem on page 54, Get-DtWorkloadPhysicalItem on page 83

Properties

Name	Type	Description
Attributes	FileSystemAttributes on page 394	This value represents the file system attributes.
CreationTime	DateTime	This value is the time when the item was created.
IsContainer	Boolean	This value is true if the item is a container of other items; otherwise, it is false.
IsReadOnly	Boolean	This value is true if the item is read-only (meaning the Saturation cannot be changed); otherwise, it is false.
ItemType	String	This value is the type of item.
LastAccessTime	DateTime	This value is the time when the item was last accessed.
LastWriteTime	DateTime	This value is the time when the item was last written to.
Metadata	String	This value is any additional metadata that may be displayed for a particular type of item. This metadata is not in any particular format, but you must be able to determine the format and decipher the metadata based on the type.
Name	String	This value is the name of the item. Typically, this name is used for display in the user interface.

Name	Type	Description
PartitionName	String	This value is the name of the partition.
Path	String	This value is the path of the item.
Saturation	SaturationLevel on page 415	This value represents the saturation level of the item. Typically, this value is used to display a visual state in the user interface.
Size	Int64	This value is the physical size of the item.

PhysicalRule

Parameter of

ChangedItems on page 222, Workload on page 365

Properties

Name	Type	Description
Inclusion	InclusionMode on page 400	This value represents if the rule will be included in or excluded from replication.
IsReadOnly	Boolean	This value is true if the rule is read-only; otherwise, it is false.
Metadata	String	This value is any additional metadata for the rule. This property can be used by specialized workload implementations to provide additional hints to the job-creation process.
Path	String	This value is the path of the data to be replicated.
Recursion	RecursionMode on page 409	This value represents if the rule will be applied to this path only or to all children of the path as well.

PhysicalVolume

Parameter of

VolumeGroup on page 355

Properties

Name	Type	Description
Attributes	FileSystemAttributes on page 394	This value represents the file system attributes.
AvailableFreeSpace	Int64	This value is the amount of free space on the volume.
CreationTime	DateTime	This value is the time when the volume was created.
DesiredSize	Int64	This value is the desired size of the new volume.
DiskControllerType	String	This value is the disk controller. • **IDE**—Use this value to create an IDE disk. • **SCSI**—Use this value to create a SCSI disk.
DiskProvisoningType	String	This value is the disk type. • **Dynamic**—Use this value for ESX thin disks and for Hyper-V dynamic disks. • **Fixed**—Use this value for ESX thick disks and for Hyper-V fixed disks. • **Flat Disk**—Use this value for ESX flat disks. This type is not supported on Hyper-V.
DriveFormat	String	This value is the file system format of the volume.
DriveType	DriveType	This value is the drive type.
IsContainer	Boolean	This value is true if the volume is a container of other volumes; otherwise, it is false.

Name	Type	Description
IsReadOnly	Boolean	This value is true if the volume is read-only; otherwise, it is false.
IsSupported	Boolean	This value is true if the volume type is supported; otherwise, it is false.
IsSystemDrive	Boolean	This value is true if the volume is the system volume; otherwise, it is false.
ItemType	String	This value is the type of item.
Label	String	This value is the time when the volume was last accessed.
LastAccessTime	DateTime	This value is the time when the volume was last accessed.
LastWriteTime	DateTime	This value is the time when the volume was last written to.
Metadata	String	This value is any additional metadata that may be displayed for a particular type of item. This metadata is not in any particular format, but you must be able to determine the format and decipher the metadata based on the type.
Name	String	This value is the name of the volume. Typically, this name is used for display in the user interface.
Path	String	This value is the path of the volume.
PreexistingDiskPath	String	This value is the full path and file name of an existing virtual disk that you want to reuse. If this value is null, a new virtual disk will be created.
Saturation	SaturationLevel on page 415	This value represents the saturation level of the volume. Typically, this value is used to display a visual state in the user interface.
ShortNameBehavior	Boolean	Do not use this property. Carbonite uses it internally.

Name	Type	Description
Size	Int64	This value is the physical size of the volume.
TotalSize	Int64	This value is the total size of the volume.
VirtualDiskPath	String	This value is the path on the host where the volume should be stored.
VolumeSignature	Int16	This value is the volume signature.
VolumeType	String	This value is the type of volume.

ProductInfo

Returned by

Get-DtProductInfo on page 55

Parameter of

CoreQualificationResults on page 238

Properties

Name	Type	Description
ActivationStatus	ActivationStatus on page 211	This value represents the activation of the server.
CanPauseTarget	Boolean	This value is true if target operations can be paused; otherwise, it is false.
CanResumeTarget	Boolean	This value is true if target operations can be resumed; otherwise, it is false.
EgineModuleStatus	Int32	Do not use this property. Carbonite uses it internally.
EnginePort	Int32	This value is the port where the engine is listening.
GatewaySessionKey	Int64	Do not use this property. Carbonite uses it internally.
InstallationPath	String	This value is the installation path.
LocalEndpoints	String	This value is the local endpoints.
MachineName	String	This value is the name of the machine.
ManagementServiceVersion	ProductVersion on page 307	This value is the version of the Management Service on the server.
Name	String	This value is the name of the product on the server.

Name	Type	Description
NatEndpoints	String	This value is the NAT endpoints.
ReservedAddress	String	This value is the reserved IP address for the server.
UniqueID	String	This value is the unique ID of the server.
Version	ProductVersion on page 307	This value is the version of the product on the server.

ProductVersion

Parameter of

ProductInfo on page 305, TargetStateInfo on page 337

Properties

Name	Type	Description
Build	Int32	This value is the build (or sequence number) of the product.
Hotfix	Int32	This value is the hotfix (or limited release) of the product.
Major	Int32	This value is the major version of the product.
Minor	Int32	This value is the minor version of the product.
ServicePack	Int32	This value is the service pack version of the product.

PSCredential

Parameter of

New-DtServer on page 103, New-DtUri on page 106, New-DtUvraServer on page 108, Set-DtJobCredentials on page 141, Set-DtScriptCredentials on page 148, Set-DtServerCredential on page 150, Test-DtActiveDirectoryCredentials on page 184, Test-DtScriptCredentials on page 190,

Properties

Name	Type	Description
UserName	String	This value is the user name.
Password	SecureString	This value is an encrypted password.

RecommendedFailbackOptions

Returned by

Get-DtRecommendedFailbackOptions on page 58

Properties

Name	Type	Description
FailbackOptions	FailbackOptions on page 256	This value represents the failback configuration.
IsSourceNew	Boolean	This value is true if the data from the target was or is being restored to a new source server; otherwise, it is false.
RestoreStatus	RestoreStatus on page 414	This value represents the status of the restoration process.

RecommendedFailoverOptions

Returned by

Get-DtRecommendedFailoverOptions on page 60

Properties

Name	Type	Description
FailoverOptions	FailoverOptions on page 257	This value represents the failover configuration.
IsTestFailoverSupported	Boolean	This value is true if test failover is supported for the job type; otherwise, it is false.
Snapshots	SnapshotEntry [ ] on page 329	This value represents the available snapshots on the target.
WarningTextTestFailover	Boolean	This value is true if optional warning text to display if test failover is selected; otherwise, it is false.
WarnUserOfInconsistentProtectionData	Boolean	This value is true if the data on the target may be in an inconsistent state; otherwise, it is false.

RecommendedJobOptions

Returned by

Get-DtRecommendedJobOptions on page 62

Properties

Name	Type	Description
JobOptions	JobOptions on page 273	This value represents the current options for the job.
JobQualificationResults	JobQualificationResults on page 275	This value represents the job qualification results which are other possible job options, in addition to those that are strictly recommended. For example, JobOptions.CoreConnectionOptions on page 235.TargetAddress may be the recommended address, but JobQualificationResults.CoreQualificationResults on page 238.TargetIPAddresses may include other IP addresses on the server.

RecommendedRestoreOptions

Returned by

Get-DtRecommendedRestoreOptions on page 65

Properties

Name	Type	Description
CanClearRestoreRequired	Boolean	This value is true if the restore required flag can be cleared; otherwise, it is false.
IsNat	Boolean	This value is true if the IP address is a public NAT address; otherwise, it is false.
PossibleSourceAddresses	String	This value is the possible source IP addresses.
RestoreOptions	RestoreOptions on page 316	This value represents the restoration options.
SameSourceOnly	Boolean	This value is true is the job can only be restored to the same source server; otherwise, it is false.

RepairStatus

Returned by

Get-DtRepairJobOptionsStatus on page 67

Properties

Name	Type	Description
Task	ActivityStatusEntry on page 212	This value represents the status of the task being repaired.
JobOptions	JobOptions on page 273	This value represents the repaired job options. This value will be null until the repair has been completed.

ReplicaVmInfo

Parameter of

VRAOptions on page 360

Properties

Name	Type	Description
Address	String	This value is the virtual machine guest name or IP address.
BiosGuid	String	This value is the virtual machine BIOS unique ID.
BootVolumeSignature	Int8 [ ]	This value is the virtual machine boot volume signature which is required for agentless Hyper-V WAN failover.
CoresPerProcessor	Int32	This value is the number of cores per processor. A value of 0 is used for an unknown or unspecified number.
Cpus	Int32	This value is the number of processors.
DisplayName	String	This value is the virtual machine display name.
GuestOS	String	This value is the virtual machine guest operating system.
GuestUri	Uri	This value is the guest URI.
Id	Guid	This value is the virtual machine unique ID.
Memory	Int64	This value is the amount of memory in bytes.
OperatingSystem	String	This value is the ESX operating system.
Path	String	This value is the virtual machine configuration file location.
PathTransformations	PathTransformation [ ] on page 298	This value represents where the protected source virtual machine will be located on the target.
PrestageFolder	String	This value is the full path to a location that contains an existing

Name	Type	Description
		virtual disk that you want to reuse.
ReplicaVmVersion	String	This value is the version of the virtual machine hardware.
RunOnceAtStartup	String	This value is a command to run when the replica virtual machine is first powered on.
SnapshotDataPath	String	This value is the virtual machine snapshot location.
SnapshotFileNames	String [ ]	This value is the virtual machine snapshot file names.
SystemDirectory	String	This value is the virtual machine system directory which is required for agentless Hyper-V WAN failover.
VirtualHardDiskPath	String [ ]	This value is the virtual machine hard disk location.

RestoreOptions

Parameter of

RecommendedRestoreOptions on page 312

Properties

Name	Type	Description
ClearRestoreRequired	Boolean	This value is true if the restore required flag can be cleared without restoring; otherwise, it is false.
EnginePort	Int32	This value is the port where the engine is listening.
RestoreParameters	RestoreParameters on page 317	This value represents the restoration configuration.
RestoreTargetHostUri	Uri	This value is the target URI. (The target is the target of the restoration connection, not the original protection connection.)

RestoreParameters

Parameter of

RestoreOptions on page 316

Properties

Name	Type	Description
ArchiveBinLocation	String	This property is no longer used.
ArchiveOption	ArchiveOption	This property is no longer used.
CompressionLevel	CompressionLevel on page 226	This value represents how data will be compressed when it is transmitted from the source to the target.
MirrorComparisonCritera	MirrorComparisonCriteria on page 402	This value represents how to compare the source protected data and the replica on the target.
OriginalSourceName	String	This value is the name of the original source server from the original protection job.
OriginalTargetRoute	String	This value is the route for the original target server from the original protection job.
PathTransformations	PathTransformation on page 298	This value represents where the source protected data will be located on the target.
ProcessOrphans	Boolean	This value is true if the orphan files (files in the target path location that are not present on the source) are to be deleted; otherwise, it is false.
ReplicationSetName	String	This value is the name of the protected data set.
RestoreOptions	RestoreParametersRestoreOptions on page 412	This value represents addtional options used for the restoration process.

ReverseOptions

Parameter of

VRAOptions on page 360

Properties

Name	Type	Description
VmName	String	This value is the display name of the reverse virtual machine.
VmPath	String	This value is the path on the host where the reverse virtual machine should be stored.
Volumes	VolumeOptions on page 356	This value represents a volume to reverse and any changes to its base configuration.
ReverseVMwareServer	URI	This value is the URI of the ESX server hosting the reverse appliance.

ScriptPoint

Parameter of

ConnectionStartParameters on page 229

Properties

Name	Type	Description
Arguments	String	This value is a comma-separated list of valid arguments required to execute the script.
ExecutionMode	ScriptExecutionMode on page 416	This value represents if Carbonite will wait while executing the script.
InteractionMode	DesktopInteractionMode on page 380	This value represents if the script processing will dispaly on screen.
Path	String	This value is the full path and file name of the script.
Type	ScriptPointType on page 417	This value represents the type of script to execute.

Server

Returned by

New-DtServer on page 103, New-DtUvraServer on page 108, Set-DtServerCredential on page 150

Parameter of

Add-DtPhysicalRule on page 16, Add-DtUvraPhysicalRule on page 18, Checkpoint-DtConnection on page 20, Close-DtWorkload on page 24, Confirm-DtJobOptions on page 25, Disconnect-DtServer on page 28, Edit-DtJob on page 29, Get-DtAccessLevel on page 31, Get-DtActivationStatus on page 32, Get-DtBandwidthLimit on page 34, Get-DtConnectionIds on page 36, Get-DtDiagnostics on page 37, Get-DtEmailNotificationOptions on page 41, Get-DtEventLogEntry on page 42, Get-DtJob on page 43, Get-DtJobActionStatus on page 45, Get-DtLogicalItem on page 48, Get-DtLogMessage on page 49, Get-DtOption on page 52, Get-DtPathBlocking on page 53, Get-DtPhysicalItem on page 54, Get-DtProductInfo on page 55, Get-DtQualificationResults on page 56, Get-DtRecommendedFailbackOptions on page 58, Get-DtRecommendedFailoverOptions on page 60, Get-DtRecommendedJobOptions on page 62, Get-DtRecommendedPathTransform on page 64, Get-DtRecommendedRestoreOptions on page 65, Get-DtScriptCredentials on page 69, Get-DtServerInfo on page 70, Get-DtSnapshot on page 71, Get-DtUvraRecommendedFailoverOptions on page 77, Get-DtUvraRecommendedRemoveOptions on page 79, Get-DtVerificationStatus on page 81, Get-DtWorkload on page 82, Get-DtWorkloadPhysicalItem on page 83, Get-DtWorkloadType on page 84, Invoke-DtQueueTask on page 92, New-DtFilesAndFoldersJob on page 98, New-DtJob on page 100, New-DtTaskParameters on page 105, New-DtWorkload on page 110, Remove-DtJob on page 112, Remove-DtPhysicalRule on page 114, Remove-DtSnapshot on page 116, Repair-DtJobOptions on page 120, Restart-DtReplicationService on page 126, Resume-DtJob on page 127, Resume-DtMirror on page 129, Resume-DtTarget on page 131, Save-DtJobDiagnostics on page 134, Set-DtActivationCode on page 136, Set-DtBandwidthLimit on page 138, Set-DtEmailNotificationOptions on page 140, Set-DtJobCredentials on page 141, Set-DtLogicalItemSelection on page 143, Set-DtOption on page 145, Set-DtPathBlocking on page 147, Set-DtScriptCredentials on page 148, Start-DtJob on page 153, Start-DtJobFailback on page 155, Start-DtJobFailover on page 157, Start-DtJobRestore on page 159, Start-DtJobReverse on page 161, Start-DtMirror on page 163, Start-DtOrphansProcessing on page 165, Start-DtReplication on page 167, Start-DtVerify on page 169, Stop-DtJob on page 171, Stop-DtMirror on page 173, Stop-DtReplication on page 175, Stop-DtReplicationService on page 177, Suspend-DtJob on page 178, Suspend-DtMirror on page 180, Suspend-DtTarget on page 182, Test-DtActiveDirectoryCredentials on page 184, Test-DtEmailNotification on page 186, Test-DtScript on page 188, Test-DtScriptCredentials on page 190, Undo-DtJobFailover on page 194, Update-DtShares on page 197, Wait-DtMirrorComplete on page 201

Properties

Name	Type	Description
Credentials	Credentials on page 240	This value represents the credentials used to access the server.

Name	Type	Description
HostName	String	This value is the name of the server.
Port	Int32	This value is the port number used to access the server.
Role	String	This value is an optional role defined for the server.
URI	String	This value is the URI of the server.

ServerActivationInformation

Returned by

Get-DtOnlineActivationRequest on page 51

Properties

Name	Type	Description
Code	String	This value is the license key.
ServerInformation	String	This value is the unique server information which will be used to generate the activation key for this particular server.
ServerName	String	This value is the name of the server.
ServiceHost	Server on page 320	This value is a server object returned from the New-DtServer cmdlet. See New-DtServer on page 103.

ServerInfo

Returned by

Get-DtServerInfo on page 70

Properties

Name	Type	Description
BiosGuid	Guid	This value is the BIOS unique ID.
BootVolume	String	This value is the server's boot volume.
CoresPerProcessor	Int32	This value is the number of cores per processor. A value of 0 is used for an unknown or unspecified number.
Disks	Disk [ ] on page 243	This value is the disks on the server.
Domain	String	This value is the domain the server is a member of.
FullyQualifiedDomain	String	This value is the fully qualified domain name.
HalInternalName	String	This value is the internal name of the hardware abstraction layer (HAL).
HalVersion	String	The version of the hardware abstraction layer
IsClustered	Boolean	This value is true if this the server is part of a cluster; otherwise, it is false.
IsHostedByHyperV	Boolean	This value is true if the server is hosted by Hyper-V; otherwise, it is false.
IsHostedByVMware	Boolean	This value is true if the server is hosted by VMware; otherwise, it is false.
IsHostedByXen	Boolean	This value is true if the server is hosted by Xen; otherwise, it is false.

Name	Type	Description
IsHyperVHost	Boolean	This value is true if the server is a Hyper-V host; otherwise, it is false.
IsReplicationEnabled	Boolean	This value is true if replication is enabled; otherwise, it is false.
IsSBS	Boolean	This value is true if the server is running Microsoft Windows Small Business Server; otherwise, it is false.
IsSSE	Boolean	This value is true if the server is running Microsoft Windows Storage Server; otherwise, it is false.
LogicalProcessorCount	Int32	This value is the number of logical processors including threads.
LvmOptions	LvmOptions on page 284	This value represents the Linux Volume Manager (LVM) options.
ManagementPort	Int32	This value is the port where the Management Service is listening.
MemrorySize	Int64	This value is the amount of memory on the server.
Name	String	This value is the name of the server.
NetworkInterfaces	NetworkInterfaceInfo [ ] on page 292	This value represents the NICs on the server.
NodeLockedServerInfo	String	This value is the server information needed for an activation key.
OperatingSystem	OperatingSystemInfo on page 293	This value is the the operating system on the server.
ProcessorCount	Int32	This value is the processor count on the server.
ProgramFilesPath	String	This value is the server's program files path.
SystemPath	String	This value is the server's system path.
SystemRoot	String	This value is the sever's system root path.

Name	Type	Description
SystemStateDefinition	String	Do not use this property. Carbonite uses it internally.
SystemVolume	String	This value is the server's system volume.
Volumes	Volume [ ] on page 353	This value represents the volumes on the server.

ServerQualificationResults

Parameter of

VRAQualificationResults on page 363

Properties

Name	Type	Description
CoresPerProcessor	Int32	This value is the number of cores per processor. A value of 0 is used for an unknown or unspecified number.
Cpus	Int32	This value is the number of processors.
LogicalProcessors	Int32	This value is the number of logical processors including threads.
Memory	Int64	This value is the amount of memory in bytes.
Version	String	This value is the version of the target host hypervisor.
VirtualSwitches	VirtualSwitchInfo [ ] on page 350	This value represents the virtual switch information
Volumes	VolumeQualificationResults on page 359	This value represents the volume information.

ServiceInformation

Parameter of

ServiceMonitoringOptions on page 328, TargetServicesOptions on page 335

Properties

Name	Type	Description
DisplayName	String	This value is the service display name.
Name	String	This value is the service name.
Selected	Boolean	This value is true if the service is selected for service monitoring; otherwise, it is false.

ServiceMonitoringOptions

Parameter of

MonitoringOptions on page 291

Properties

Name	Type	Description
RepeatCount	Int32	This value is the number of times to repeat the call to retrieve the service status.
Services	ServiceInformation [ ] on page 327	This value represents the services that should be monitored.
StartService	Boolean	This value is true if the monitored service should be started if it is stopped; otherwise, it is false.

SnapshotEntry

Returned by

Get-DtSnapshot on page 71

Parameter of

RecommendedFailoverOptions on page 310, TargetStateInfo on page 337

Properties

Name	Type	Description
Attributes	SnapshotAttributes on page 419	This value represents a snapshot created by the Windows Volume Snapshot Service.
Comment	String	This value is any description associated with the snapshot.
ConName	String	This value is the connection name for the snapshot.
Id	Guid	This value is the unique ID for the snapshot.
Reason	SnapshotCreationReason on page 420	This value represents why the snapshot was taken.
States	TargetStates on page 424	This value represents the state of the target associated with the snapshot.
Timestamp	DateTimeOffset	This value is the time when the snapshot was taken.

SnapshotSchedule

Parameter of

ConnectionStartParameters on page 229

Properties

Name	Type	Description
Interval	TimeSpan	This value is the interval for taking snapshots.
IsEnabled	Boolean	This value is true if the snapshot schedule is set; otherwise, it is false.
MaxNumberOfSnapshots	Int32	This value is the maximum number of snapshots that will be retained.
StartTime	DateTime	This value is the when the snapshot schedule should start.

SourceQueueSnapshotEntry

Returned by

Get-DtSourceQueueSnapshot on page 73, Get-DtSourceQueueSnapshots on page 75

Properties

Name	Type	Description
CorrelationId	Guid	This value is the correlation ID for the snapshot.
JobId	Guid	This value is the job ID.
Quality	SnapshotQuality on page 421	This value represents the quality of the snapshot.
Reason	SnapshotCreationReason on page 420	This value represents why the snapshot was taken.
SnapshotSetId	Guid	This value is the snapshot ID.
SourceAddress	String	This value is the IP address of the source server
State	SnapshotState on page 422	This value is the state of the snapshot.
TargetStates	TargetStates on page 424	This value represents the state of the target associated with the snapshot.
TimeStampCompleted	DateTimeOffset	This value is when the coordinated snapshot was completed.
TimeStampRequested	DateTimeOffset	This value is when the coordinated snapshot was requested.

SystemStateOptions

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
AlternateVolumeMapping	String	This value is the mapping of source volumes to target volumes for alternate volume staging, for example, C:;N:.
AlternateVolumeStaging	Boolean	This value is true if volumes will be staged to an alternate volume; otherwise it is false.
ApplyPorts	Boolean	This value indicates if the source ports should be copied to the target.
ClearMonitor	Boolean	This value is true if the failover monitor should be removed at cutover; otherwise, it is false.
IsWanFailover	Boolean	This value is true if the job uses WAN failover; otherwise,it is false.
KeepTargetActivationCode	Boolean	This value is true if the target license key should be kept after failover; otherwise, it is false (and the source's license key will be applied to the target after failover).
NicMappings	FullServerNicMappings [ ] on page 265	This value represents the NIC mappings for a full server job.
ServerNameOverride	String	Do not use this property. Carbonite uses it internally.
ServicesToStopOptions	TargetServicesToStop [ ] on page 336	This value represents the services on the target to stop during protection.
ShouldApplyDiskSignatures	Boolean	This value is true if disk signatures should be applied at cutover; otherwise, it is false.

Name	Type	Description
SourceReservedAddress	String	This value is the reserved IP address on the source server.
StagingFolder	String	This value is the folder on the target where operating system files from the source will be staged.
TargetReservedAddress	String	This value is the reserved IP address on the target server

TargetFileServerQualificationResults

Parameter of

ClusterFilesAndFoldersQualifcationResults on page 224

Properties

Name	Type	Description
ClusterResourceGroupIPAddresses	UnicastIPAddressInfo [ ] on page 343	This value represents a unicast IP address.
ClusterResourceGroupName	String	This value is the name of the cluster group that contains the resource.
CurrentOwnerNodeName	String	This value is the name of the resource's current owning node.
DiskSize	Int64	This value is total disk size, in bytes, of the volume.
DriveLetter	String	This value is the volume drive letter.
FreeSpace	Int64	This value is the free space, in bytes, available on the volume.
IsVolumeCSV	Boolean	This value is true if it is a cluster shared volume; otherwise, it is false.
RecommendedGroup	Boolean	This value is true if there is a file server group on the target cluster that matches the protected file server group from the source; otherwise, it is false. When this value is true, an IP address from the group will be used as the route for CoreConnectionOptions on page 235.TargetAddress.

TargetServicesOptions

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
FailoverServices	ServiceInformation [ ] on page 327	This value represents services that will be stopped on the source and started on the target during failover and started on the source and stopped on the target during failback.
StartAndStopServices	Boolean	This value is true if the services will be stopped and started; otherwise, it is false (and the services will be left running).

TargetServicesToStop

Parameter of

SystemStateOptions on page 332

Properties

Name	Type	Description
Failover	Boolean	This value is true if the service should be started after failover; otherwise, it is false. In addition to starting this service after failover, this value is whether the service will be stopped during failback.
IsCritical	Boolean	This value is true if the service is considered critical; otherwise, it is false.
KeepRunningNonCritical	Boolean	This value is true if the non-critical service should be kept running; otherwise, it is false.
ServiceDescription	String	This value is the service description.
ServiceName	String	This value is the name of the service
State	TargetServiceStatus on page 423	This value represents the state of a service.

TargetStateInfo

Parameter of

UnmanagedConnectionOptions on page 344

Properties

Name	Type	Description
ConnectionId	Guid	This value is the unique ID assigned to the connection.
ConnectTime	DateTimeOffset	This value is the time when the connection was created.
EngineConnectionId	Int32	This value is the unique ID assigned to the replication engine connection.
EngineJobType	EngineJobType on page 382	This value is the job type according to the replication engine. (These jobs are not the same as JobInfo on page 270.JobType.)
HasSnapshotSchedule	Boolean	This value is true if the connection has a snapshot schedule; otherwise, it is false.
LastUpdateTime	DateTimeOffset	This value is the last time the target connection information was updated.
NextScheduledSnapshot	DateTimeOffset	This value is the time of the next scheduled snapshot. This property is only valid when HasSnapshotSchedule is true.
Paths	String [ ]	This value is the paths of the replica data on the target.
QueueBytes	Int64	This value is the number of bytes in the target queue.
ReplicationSetName	String	This value is the name of the protected data set.
ReplicationSetUsageType	ReplicationSetUsageType on page 410	This value defines the possible usage types for the protected data set.

Name	Type	Description
ScheduledSnapshotInterval	TimeSpan	This value is the interval for the scheduled snapshot. This property is only valid when HasSnapshotSchedule is true.
Snapshots	SnapshotEntry [ ] on page 329	This value represents the current list of snapshots for the connection.
SourceEndpoint	String	This value is the IP endpoint used to communicate with Carbonite on the source.
SourceEndpointFromSource	String	This value is the default address for the source server as provided by the source server.
SourceMachineName	String	This value is the name of the source server.
SourceVersion	ProductVersion on page 307	This value is the Carbonite product version on the source.
TargetEndpoint	String	This value is the destination endpoint of the socket connection. If there is a NAT router between the source and target, this value represents the target's private address (translated from the public address). In other words, this will not match the target endpoint specified for the target route of the connection from the source's perspective.
TargetStates	TargetStates on page 424	This values is the state of the target.

TaskParameters

Returned by

New-DtTaskParameters on page 105

Parameter of

Invoke-DtQueueTask on page 92

Properties

Name	Type	Description
Arguments	String	This value is a comma-separated list of valid arguments required to execute the script.
Script	String	This value is the full path and file name of the script.

TestFailoverOptions

Parameter of

VRAOptions on page 360

Properties

Name	Type	Description
DeleteSnapshots	Boolean	This value is true if snapshots will be deleted after the test failover; otherwise, it is false.
DeleteVirtualDisks	Boolean	This value is true if the virtual disks will be deleted after the test failover; otherwise, it is false.
DiskOptions	DiskOptions [ ] on page 244	This value is the disk options available on a Linux server. Use this property for the match source disk configuration strategy. If you want a per volume disk configuration strategy use VolumeOptions on page 356.
LvmOptions	LvmOptions on page 284	This value represents logical volume manager (LVM) options for a Linux server.
ReplicaDisplayName	String	This value is the replica display name of the test failover server. This value is only for full server to ESX for Linux jobs and allows you to specify an alternate server for test failover.
Volumes	VolumeOptions [ ] on page 356	This value represents the test volume and any changes to its base configuration. For Linux, use this property for the per volume disk configuration strategy. If you want a matching source disk configuration strategy use DiskOptions on page 244.

TestFailoverServerCredentials

Parameter of

FullServerTestFailoverOptions on page 266

Properties

Name	Type	Description
TestFailoverServerHardwareId	String	This value is the hardware ID of the test server for full server jobs.
TestFailoverServerHostUri	URI	This value is the URI of the test server for full server jobs.

TimeClass

Parameter of

FailoverReport on page 258

Properties

Name	Type	Description
completed	DateTime	This value is the time when failover completed.
startProcessing	DateTime	This value is the time when failover processing started.
startReboot	DateTime	This value is the time when the failover reboot started. For files and folders jobs, this value will be null.

UnicastIPAddressInfo

Parameter of

CoreQualificationResults on page 238, NetworkInterfaceInfo on page 292, TargetFileServerQualificationResults on page 334, VirtualNetworkInterfaceInfo on page 348

Properties

Name	Type	Description
IPAddress	String	This value is the IP address as a string.
IPv4Mask	String	This value is the the IPv4 subnet mask.
IsDHCP	Boolean	This value is true if the IP address is a DHCP assigned address; otherwise, it is false.
IsNAT	Boolean	This value is true if the IP address is a public NAT address; otherwise, it is false.
IsOnline	Boolean	This value is true if the IP address is online; otherwise, it is false.

UnmanagedConnectionOptions

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
TargetStateInfo	TargetStateInfo on page 337	This value represents information on the state of the target.

VerificationStatus

Returned by

Get-DtVerificationStatus on page 81

Properties

Name	Type	Description
Steps	VerificationStep [ ] on page 346	This value represents a step in the job validation process. Do not confuse this process with the verification process that confirms if the data between the source and target are synchronized.
Task	ActivityStatusEntry on page 212	This value represents the status of the validation task.

VerificationStep

Returned by

Wait-DtConfirmJobOptions on page 199

Parameter of

Repair-DtJobOptions on page 120, VerificationStatus on page 345

Properties

Name	Type	Description
CanFix	Boolean	This value is true if Carbonite can automatically fix the validation item; otherwise, it is false.
Id	String	This value is the ID for the validation item.
Index	Int32	This value is an index value used to impose an order to the validation items.
MessageFormatParameters	String	This value is the message format parameters.
MessageKey	String	This value is a message key for the validation item. This value can be used to look up the message text.
Progress	Int32	This value is the completion progress of the validation item.
Status	ActivityCompletionStatus on page 375	This value represents the status of the validation item.
TitleFormatParameters	String [ ]	This value is the title format parameters for the action.
TitleKey	String	This value is a title key for the validation item. This value can be used to look up the title text.

VerifySchedule

Parameter of

ConnectionSchedule on page 228

Properties

Name	Type	Description
Criteria	MirrorComparisonCriteria on page 402	This value represents how to compare the source protected data and the replica on the target.
Interval	TimeSpan	This value is the interval for taking snapshots.
IsEnabled	Boolean	This value is true if the verification schedule is set; otherwise, it is false.
Options	MirrorOperationOptions on page 403	This value represents what mirror operation will be performed.
StartTime	DateTime	This value is the when the verification schedule should start.

VirtualNetworkInterfaceInfo

Parameter of

VRAOptions on page 360

Properties

Name	Type	Description
Description	String	This value is the description of the NIC.
DnsDomain	String	This value is the DNS domain assigned to the NIC.
DnsServers	String	This value is the DNS server assigned to the NIC.
Gateways	String	This value is the gateway assigned to the NIC.
Guid	String	This value is the unique ID for the NIC. It is equivalent to the SettingID property of the Win32_NetworkAdapterConfiguration WMI class as well as the NetworkInterface.Id property in the Base Class Library (BCL).
Index	Int32	Do not use this property. Carbonite uses it internally.
InterfaceIndex	Int32	Do not use this property. Carbonite uses it internally.
IPAddresses	UnicastIPAddressInfo [ ] on page 343	This value represents the IP addresses assigned to the NIC.
MacAddresses	String	This value represents the MAC address of the NIC.
Name	String	This value is the friendly name of the NIC.
PnpInstanceId	String	This value is the plug and play instance ID of the NIC.
ServiceName	String	This value is the service name of the NIC.
VirtualNetwork	String	This value is the virtual network.
VirtualNictype	String	This value is one of the following virtual NIC types.

Name	Type	Description
		• Default • Legacy • Synthetic • E1000 • PCNet32 • VmxNet • VmxNet3 • xennet
VLAN_ID	Int32	This value is the ID of the VLAN on the replica after live failover. The value can be any integer between 1 and 4095, or a value of 0 indicates the server does not support setting the VLAN ID, or a value of -1 indicates the VLAN ID should not be set. No other values are supported.
VLAN_ID_TestFailover	Int32	This value is the ID of the VLAN on the replica after test failover. The value can be any integer between 1 and 4095, or a value of 0 indicates the server does not support setting the VLAN ID, or a value of -1 indicates the VLAN ID should not be set. No other values are supported.

VirtualSwitchInfo

Parameter of

ServerQualificationResults on page 326, VirtualSwitchMapping on page 351

Properties

Name	Type	Description
Label	String	This value is the virtual switch label.
SwitchUuid	String	This value is the virtual switch unique ID.

VirtualSwitchMapping

Parameter of

VRAOptions on page 360

Properties

Name	Type	Description
SourceVirtualSwitch	VirtualSwitchInfo on page 350	This value represents the virtual switch on the source.
TargetVirtualSwitch	VirtualSwitchInfo on page 350	This value represents the virtual switch on the target.

VmInfo

Parameter of

VRAOptions on page 360

Properties

Name	Type	Description
Address	String	This value is the virtual machine guest name or IP address.
BiosGuid	Guid	This value is the virtual machine BIOS unique ID.
BootVolumeSignature	Int8 [ ]	This value is the virtual machine boot volume signature which is required for agentless Hyper-V WAN failover.
DisplayName	String	This value is the virtual machine display name.
EsxHost	String	This value is the ESX server hosting the virtual machine.
GuestOS	String	This value is the virtual machine guest operating system.
GuestUri	Uri	This value is the guest URI.
Id	Guid	This value is the virtual machine unique ID.
Path	String	This value is the virtual machine configuration file location.
PathTransformations	PathTransformation [ ] on page 298	This value represents where the protected source virtual machine will be located on the target.
SnapshotDataPath	String	This value is the virtual machine snapshot location.
SnapshotFileNames	String [ ]	This value is the virtual machine snapshot file names.
SystemDirectory	String	This value is the virtual machine system directory which is required for agentless Hyper-V WAN failover.
VirtualHardDiskPath	String [ ]	This value is the virtual machine hard disk location.

Volume

Parameter of

CoreQualificationResults on page 238, ServerInfo on page 323

Properties

Name	Type	Description
Attributes	FileSystemAttributes on page 394	This value represents the file system attributes.
AvailableFreeSpace	Int64	This value is the amount of free space on the volume.
CreationTime	DateTime	This value is the time when the item was created.
DriveFormat	String	This value is the file system format of the volume.
DriveType	DriveType	This value is the drive type.
IsContainer	Boolean	This value is true if the volume is a container of other volumes; otherwise, it is false.
IsReadOnly	Boolean	This value is true if the volume is read-only; otherwise, it is false.
IsSupported	Boolean	This value is true if the volume type is supported; otherwise, it is false.
IsSystemDrive	Boolean	This value is true if the volume is the system volume; otherwise, it is false.
ItemType	String	This value is the type of item.
Label	String	This value is the label assigned to the volume.
LastAccessTime	DateTime	This value is the time when the volume was last accessed.
LastWriteTime	DateTime	This value is the time when the volume was last written to.

Name	Type	Description
Metadata	String	This value is any additional metadata that may be displayed for a particular type of item. This metadata is not in any particular format, but you must be able to determine the format and decipher the metadata based on the type.
Name	String	This value is the name of the volume. Typically, this name is used for display in the user interface.
PartitionName	String	This value is the name of the partition.
Path	String	This value is the path of the volume.
Saturation	SaturationLevel on page 415	This value represents the saturation level of the volume. Typically, this value is used to display a visual state in the user interface.
ShortNameBehavior	Boolean	Do not use this property. Carbonite uses it internally.
Size	Int32	This value is the physical size of the volume.
TotalSize	Int64	This value is the total size of the volume.
VolumeType	String	This value is the type of volume.

VolumeGroup

Parameter of

LvmOptions on page 284

Properties

Name	Type	Description
LogicalVolume	LogicalVolume [ ] on page 280	This value represents the logical volumes in the volume group.
MaxPhysicalVolumeSize	Int64	This value is the maximum size, in bytes, of the virtual disks used to create the volume group. The default value is equal to the maximum size that can be attached to the datastore you selected. That will depend on your ESX version, your file system version, and the block size of your datastore.
Name	String	This value is the name of the volume group on the source.
PhysicalVolume	PhysicalVolume [ ] on page 302	This value represents the physical volumes in the volume group.
PreexistingDisksPath	String [ ]	This value is the full path and file name of an existing virtual disk that you want to reuse. If this value is null, a new virtual disk will be created.
SourceVolumeGroupSize	Int64	This value is the size of the volume group on the source.

VolumeOptions

Parameter of

ReverseOptions on page 318, TestFailoverOptions on page 340, VRAOptions on page 360

Properties

Name	Type	Description
Attributes	FileSystemAttributes on page 394	This value represents the file system attributes.
AvailableFreeSpace	Int32	This value is the amount of free space on the volume.
CreationTime	DateTime	This value is the time when the volume was created.
DesiredSize	Int64	This value is the desired size of the new volume.
DiskControllerType	String	This value is the disk controller. • **IDE**—Use this value to create an IDE disk. • **SCSI**—Use this value to create a SCSI disk.
DiskProvisioningType	String	This value is the disk type. • **Dynamic**—Use this value for ESX thin disks and for Hyper-V dynamic disks. • **Fixed**—Use this value for ESX thick disks and for Hyper-V fixed disks. • **Flat Disk**—Use this value for ESX flat disks. This type is not supported on Hyper-V.
DriveFormat	String	This value is the file system format of the volume.
DriveType	DriveType	This value is the drive type.
Guid	Guid	This value is the UUID assigned to an existing virtual disk.
IsContainer	Boolean	This value is true if the volume is a container of other volumes;

Name	Type	Description
		otherwise, it is false.
IsReadOnly	Boolean	This value is true if the volume is read-only; otherwise, it is false.
IsSupported	Boolean	This value is true if the volume type is supported; otherwise, it is false.
IsSystemDrive	Boolean	This value is true if the volume is the system volume; otherwise, it is false.
ItemType	String	This value is the type of item.
Label	String	This value is the label assigned to the volume.
LastAccessTime	DateTime	This value is the time when the volume was last accessed.
LastWriteTime	DateTime	This value is the time when the volume was last written to.
Metadata	String	This value is any additional metadata that may be displayed for a particular type of item. This metadata is not in any particular format, but you must be able to determine the format and decipher the metadata based on the type.
Name	String	This value is the name of the volume. Typically, this name is used for display in the user interface.
PartitionName	String	This value is the name of the partition.
Path	String	This value is the path of the volume.
PreexistingDiskPath	String	This value is the full path and file name of an existing virtual disk that you want to reuse. If this value is null, a new virtual disk will be created.
Saturation	SaturationLevel on page 415	This value represents the saturation level of the volume. Typically, this value is used to display a visual state in the user interface.

Name	Type	Description
ShortNameBehavior	Boolean	Do not use this property. Carbonite uses it internally.
Size	Int32	This value is the physical size of the volume.
TotalSize	Int32	This value is the total size of the volume.
VirtualDiskPath	String	This value is the path on the host where the volume should be stored.
VolumeSignature	Int8	This value is the volume signature.
VolumeType	String	This value is the type of volume.

VolumeQualificationResults

Parameter of

ServerQualificationResults on page 326

Properties

Name	Type	Description
ClusterResourceGroupName	String	This value is the name of the group that contains the resource.
CurrentOwnerNodeName	String	This value is the name of the resource's current owning node.
DiskSize	Int64	This value is the total size, in bytes, of the volume.
DriveLetter	String	This value is the drive letter of the volume.
FreeSpace	Int64	This value is the available free space, in bytes, on the volume.
IsSystemVolume	Boolean	This value is true if the volume is the system volume; otherwise, it is false.
IsVolumeCSV	Boolean	This value is true if the volume is a cluster shared volume; otherwise, it is false.
MaxFileSize	Int64	This value is the maximum file size, in mebabytes, of the virtual hard disk file.
ProvisionedSpace	Int64	This value is the amount of provisioned space, in bytes, on the volume.
Url	String	This value is the URL of an ESX datastore.

VRAOptions

Parameter of

JobOptions on page 273

Properties

Name	Type	Description
Hypervisor	String	This value should be one of the following strings. • VMWARE.ESX • VCloud.ESX • Windows.HyperV • Amazon.XEN • Citrix.XEN
IsSourceHostCluster	Boolean	This value is true if the source host is clustered; otherwise, it is false.
IsWanFailoverEnabled	Boolean	This value is true if WAN failover is enabled; otherwise, it is false.
DiskOptions	DiskOptions [ ] on page 244	This value is the disk options available on a Linux server. Use this property for the match source disk configuration strategy. If you want a per volume disk configuration strategy use VolumeOptions on page 356.
DiskConfigStrategy	DiskConfigStrategy on page 381	This value represents the disk configuration strategy used on the target server.
LvmOptions	LvmOptions on page 284	This value represents logical volume manager (LVM) options for a Linux server.
ReplicaApplianceInfo	VmInfo on page 352	This value represents the helper appliance information.

Name	Type	Description
ReplicaESXHostName	String	This value is the name of the ESX host where the replica will be located.
ReplicaNetworkInterfaceInfo	VirtualNetworkInterfaceInfo [ ] on page 348	This value represents the desired NIC configuration on the replica virtual machine.
ReplicaVmInfo	ReplicaVmInfo on page 314	This value represents the desired configuration of the replica virtual machine.
ReverseCount	Int32	Do not use this property. Carbonite uses it internally.
ReverseRoute	String	This value is the source IP address to be used for the reverse connection.
ReverseOptions	ReverseOptions on page 318	This value represents the options used for the reverse process.
SourceApplianceInfo	VmInfo on page 352	This value represents the appliance used during reverse.
SourceESXHostName	String	This value is the name of the ESX host where the source is located.
SourceHypervisor	String	This value should be one of the following strings. • VMWARE.ESX • VCloud.ESX • Windows.HyperV • Amazon.XEN • Citrix.XEN
SourceNetworkInterfaceInfo	VirtualNetworkInterfaceInfo [ ] on page 348	This value represents the available NICs on the source.
SourceProductLicense	String	This value is the license key used for automatic installations during the V to ESX and V to Hyper-V job creation process.
SourceVmInfo	VmInfo on page 352	This value represents the source information used during

Name	Type	Description
		reverse.
TestFailover	TestFailoverOptions on page 340	This values represents the test failover options for full server to ESX and full server to Hyper-V jobs.
VirtualSwitchMapping	VirtualSwitchMapping [ ] on page 351	This value represents the virtual switch mappings on the source and target for live failover.
VirtualSwitchMappingTestFailover	VirtualSwitchMapping [ ] on page 351	This value represents the virtual switch mappings on the source and target for test failover.
Volumes	VolumeOptions [ ] on page 356	This value represents a volume to protect and any changes to its base configuration. For Linux, use this property for the per volume disk configuration strategy. If you want a matching source disk configuration strategy use DiskOptions on page 244.
WorkloadCustomizationOptions	VRAWorkloadCustomizationOptions on page 364	This value represents workload customization options.

VRAQualificationResults

Parameter of

JobQualificationResults on page 275

Properties

Name	Type	Description
PreexistingDisksFileName	String	This value is the full path and file name of an existing virtual disk that you want to reuse. If this value is null, a new virtual disk will be created.
ReversetHost	ServerQualificationResults on page 326	This value represents the reverse server information.
SourceServerCoresPerProcessorCount	Int32	This value is the number of cores per processor. A value of 0 is used for an unknown or unspecified number.
SourceServerMemorySize	Int64	This value is the amount of memory, in bytes, on the source.
SourceServerProcessorCount	Int32	This value is the number of processors on the source.
TargetHost	ServerQualificationResults on page 326	This value represents the target server information.
V2VQualificationResults	None	This property is no longer used.

VRAWorkloadCustomizationOptions

Parameter of

VRAOptions on page 360

Properties

Name	Type	Description
NoReplication	Boolean	Do not use this property. Carbonite uses it internally.
PowerupReplicaAfterFailover	Boolean	This value is true if the replica should be powered on after failover; otherwise, it is false.
ShouldShutdownSource	Boolean	This value is true if the source should be shut down during failover; otherwise, it is false.
UseWin32	Boolean	Do not use this property. Carbonite uses it internally.

Workload

Returned by

Add-DtUvraPhysicalRule on page 18, Get-DtWorkload on page 82

Parameter of

Add-DtUvraPhysicalRule on page 18, Get-DtRecommendedJobOptions on page 62, JobOptions on page 273, New-DtWorkload on page 110

Properties

Name	Type	Description
LogicalRules	String [ ]	This value is the logical replication rules that define the workload.
PhysicalRules	PhysicalRule [ ] on page 301	This value represents the physical replication rules that define the workload.
RecoveryImageDataPath	None	This property is no longer used.
RecoveryImageId	None	This property is no longer used.
RecoverySnapshotSetId	None	This property is no longer used.

WorkloadTypeName	String	This value is the workload type name.
		• **Availability for Windows jobs**
		  • **FilesAndFolders**—Files and folders
		  • **ClusterAwareFilesAndFolders**—Cluster-aware files and folders
		  • **SQL**—SQL
		  • **ClusterAwareSql**—Cluster-aware SQL
		  • **FullServerFailover**—Full server
		  • **VRA**—Full server to ESX or full server to Hyper-V
		• **Availability for Linux jobs**

Name	Type	Description
		• **LinuxFullServerFailover**—Full server • **Lvra**—Full server to ESX • **Migrate for Windows jobs** • **MoveDataOnlyMigration**—Files and folders migration • **MoveServerMigration**—Full server migration • **VraMove**—Full server to ESX migration or full server to Hyper-V migration • **Migrate for Linux jobs** • **LinuxMoveServerMigration**—Full server migration • **MoveLvra**—Full server to ESX migration • **Other jobs** • **Diagnostics**—Throughput Diagnostic Utility

WorkloadSupportSummary

Parameter of

WorkloadType on page 368

Properties

Name	Type	Description
Reason	String	This value can be a string resource ID used for obtaining the reason text in the client or the reason text itself. If the reason text needs to be formatted with parameters, use ReasonFormatParameters.
ReasonFormatParameters	String	This value is format parameters for Reason.

WorkloadType

Returned by

Get-DtWorkloadType on page 84

Properties

Name	Type	Description
IsLicensed	Boolean	This value is true if this type of workload is licensed for the server; otherwise, it is false.
IsPresent	Boolean	This value is true if this type of workload is present on the server; otherwise, it is false.

Name	String	This value is the workload type name.
		• **Availability for Windows jobs**
		  • **FilesAndFolders**—Files and folders
		  • **ClusterAwareFilesAndFolders**—Cluster-aware files and folders
		  • **SQL**—SQL
		  • **ClusterAwareSql**—Cluster-aware SQL
		  • **FullServerFailover**—Full server
		  • **VRA**—Full server to ESX or full server to Hyper-V
		• **Availability for Linux jobs**
		  • **LinuxFullServerFailover**—Full server
		  • **Lvra**—Full server to ESX
		• **Migrate for Windows jobs**
		  • **MoveDataOnlyMigration**—Files and folders migration
		  • **MoveServerMigration**—Full server migration
		  • **VraMove**—Full server to ESX migration or full

Name	Type	Description

		server to Hyper-V migration
		• **Migrate for Linux jobs**
		  • **LinuxMoveServerMigration**—Full server migration
		  • **MoveLvra**—Full server to ESX migration
		• **Other jobs**
		  • **Diagnostics**—Throughput Diagnostic Utility

SupportSummary	WorkloadSupportSummary on page 367	This value is a summary of the licensing for the workload type. The summary can be null if there is no reason to report any details about the license calculation. Otherwise, it should be non-null and populated with details about the license calculation that the client can use for reporting purposes.

Chapter 4 Enumerations

The following enumerations are used in the Carbonite classes.

AccessLevel

Returned by

Get-DtAccessLevel on page 31

Parameter of

CoreConnectionDetails on page 231

Properties

Name	Enumeration	Description
Unknown	-1	The access level is unknown.
NoAccess	0	The credentials are not authenticated.
MonitorOnlyAccess	1	This access level is associated with membership in the Double-Take Monitors or dtmon security group.
FullAccess	2	This access level is associated with membership in the Double-Take Admin or dtadmin security group.

ActionStatus

Parameter of

JobAction on page 269

Properties

Name	Enumeration	Description
Pending	0	The action has not yet started.
Running	1	The action is running.
Completed	2	The action has completed successfully.
Cancelled	3	The action was canceled.
Faulted	4	The action encountered an error.

ActiveDirectoryFailoverOptions

Parameter of

MonitorConfiguration on page 287

Properties

Name	Enumeration	Description
None	0	There are no Active Directory options applied.
FailoverHostName	1	The host name of the source server should be moved to the Active Directory server object of the target when a failover occurs.
FailbackHostName	2	The host name of the target server should be reinstated in the Active Directory server object of the target when failback occurs.

ActivityCompletionStatus

Parameter of

ActivityStatusEntry on page 212, VerificationStep on page 346

Properties

Name	Enumeration	Description
Pending	0	The activity is pending.
Running	1	The activity is running.
Completed	2	The activity has completed successfully.
Canceled	3	The activity was canceled.
Faulted	4	The activity had an error.

BandwidthEntryType

Parameter of

BandwidthEntry on page 216

Properties

Name	Enumeration	Description
Daytime	0	This value is an entry where the BandwidthEntry on page 216.StartTime and .EndTime are on the same day.
Overnight	1	This value is an entry where the BandwidthEntry on page 216.StartTime value is on one day and the .EndTime value is on the next day.

BandwidthScheduleMode

Parameter of

BandwidthLimit on page 217, BandwidthOptions on page 218, BandwidthSchedule on page 219

Properties

Name	Enumeration	Description
NotLimited	0	Bandwidth is not limited.
Fixed	1	Bandwidth is limited to a fixed value.
Scheduled	2	Bandwidth is limited according to scheduled values.

BandwidthSpecificationType

Parameter of

BandwidthSpecification on page 221

Properties

Name	Enumeration	Description
LAN	0	This value is the bandwidth associated with local area networks.
WAN	1	This value is the bandwidth associated with wide area networks.

ClusterResourceState

Parameter of

CoreConnectionDetails on page 231

Properties

Name	Enumeration	Description
Uninitialized	0	A cluster resource is not used.
OfflinePending	1	The resource state changed to offline pending.
Offline	2	The resource state changed to offline.
OnlinePending	3	The resource state changed to online pending.
Online	4	The resource state changed to online.
ResourceNotFound	5	The cluster resource was not found on the cluster.
Reconnected	6	The cluster resource reconnected. This could be due to a node roll.
Connected	7	The cluster resource is connected and online.

DesktopInteractionMode

Parameter of

ScriptPoint on page 319, Test-DtScript on page 188

Properties

Name	Enumeration	Description
None	0	The script will execute silently in the background.
Interact	1	Script processing will be displayed on screen.

DiskConfigStrategy

Parameter of

VRAOptions on page 360

Properties

Name	Enumeration	Description
CreateDiskPerVolume	0	The target disk configuration is per source volume.
MatchingSource	1	The target disk configuration matches the source disk configuration.
Customization	2	The target disk configuration is a custom configuration.

EngineJobType

Parameter of

CloudOptions on page 223, TargetStateInfo on page 337

Properties

Name	Enumeration	Description
NormalJob	0	This value is a files and folders connection.
ImageJob	1	This property is no longer used.
RecoveryJob	2	This property is no longer used.
FullServerJob	4	This value is a full server connection for Windows.
GeoClusterJob	8	This property is no longer used.
MigrationJob	16	This value is a Carbonite Migrate connection.
FullServerRevertJob	32	This value is a full server reverse connection.
VraRecoveryJob	64	This value is a full server to ESX or Hyper-V reverse connection.
VraMigrationJob	128	This value is a full server to ESX or Hyper-V Carbonite Migrate connection.
DataOnlyOption	256	This value is for a connection that is replicating data only.
VraJob	512	This value is a full server to ESX or Hyper-V Carbonite Availability connection.
HyperVJob	1024	This value is a Hyper-V connection.
Win32MirrorOption	2048	This value is a connection that will override the server default and use the Win32 mirroring driver.

Name	Enumeration	Description
SourceConnectionResourceJob	4096	This value is a connection protecting a source cluster.
FullServerBackupJob	8192	This value is a full server reverse connection.
UvraJob	16384	This value is a full server to ESX appliance job.
NonClientAccessibleSnapshot	32768	This value indicates that snapshots created for this connection should not have the VSS_CTX_CLIENT_ACCESSIBLE context.
Invalid	65535	This value is an unknown connection.
FFO	65536	This value is a full server connection for Windows. This is not the real value for engine job type, but is for CreateConnectionAction to create a connection.
LVRA_JOB	131072	This value is a full server to ESX coonection for Linux. It is usually a connection to the local server (the source is same as the target) where the target drive is an iSCSI hosted by a Linux appliance. This is not the real value for engine job type, but is for CreateConnectionAction to create connection.

FailoverDataAction

Parameter of

FailoverOptions on page 257, MonitorConfiguration on page 287

Properties

Name	Enumeration	Description
Apply	0	This value will apply the data in the target queue before starting failover.
Flush	1	This value will discard the data in the target queue and start failover immediately.
Revert	2	If the target data is in a bad state, this value will revert to the last good snapshot, if snapshots are available. If the target data is in a good state or no snapshots are available, this value will apply the data in the target queue and then failover.
Unknown	3	The action is unknown.

FailoverIPAddressesOption

Parameter of

MonitorConfiguration on page 287

Properties

Name	Enumeration	Description
Monitored	0	Failover only the IP addresses that are being monitored.
All	1	Failover all of the IP address on the source.

FailoverItems

Parameter of

MonitorConfiguration on page 287

Properties

Name	Enumeration	Description
None	0	Nothing will be failed over.
IPAddresses	1	The IP addresses specified by MonitorConfiguration on page 287.FailoverIPAddressesOption will be applied to the target during failover.
Name	2	The NetBIOS name of the source server will be applied to the target during failover.
Shares	4	Shares included in the source workload will be added to the target during failover.

FailoverMode

Parameter of

FailoverOptions on page 257

Properties

Name	Enumeration	Description
Live	0	Failover is using live, current data and the target is started with network connectivity.
Test	1	Failover is using live current data, but is a test. The target is started without network connectivity.
Snapshot	2	Failover is using a data from a snapshot. The target is started with network connectivity.
SetRestoreRequiredOnly	3	Failover is not performed, but the restore required fiag is set.

FailoverProcessingOptions

Parameter of

MonitorConfiguration on page 287

Properties

Name	Enumeration	Description
None	0	This value resets the flags so no failover processing options will be used.
UserInterventionRequired	1	The user must manually initiate failover when a failover condition is met.
UseShareFile	2	When failing over file shares, the persisted share file on the target should be used to create the shares on the target. Without this value, the target will attempt to obtain the shares from the source during failover, however, if the source is unavailable, no shares will be created on the target.

FailoverReplaceActions

Parameter of

MonitorConfiguration on page 287

Properties

Name	Enumeration	Description
None	0	This value resets the flags so no failover replace actions will be used.
Name	1	The NetBIOS name of the target will be replaced by the NetBIOS name from the source during failover.
Address	2	The IP addresses on the target will be replaced by the IP address from the source during failover.

FailoverStyle

Parameter of

FailoverReport on page 258

Properties

Name	Enumeration	Description
Live	0	Failover is using live, current data
Test	1	Failover is using live current data, but is a test.

FailoverTrigger

Parameter of

MonitorConfiguration on page 287

Properties

Name	Enumeration	Description
OneAddressFails	0	A failover condition is met when one monitored IP address fails.
AllAddressesFail	1	A failover condition is met when all monitored IP addresses fail.

FailoverType.Monitor

Parameter of

MonitorConfiguration on page 287

Properties

Name	Enumeration	Description
Normal	0	The parameters in MonitorConfiguration on page 287 are used to process failover.
FullServer	1	The entire server is being failed over and internal Carbonite processing will determine what is failed over.

FailoverType.Options

Parameter of

FailoverOptions on page 257

Properties

Name	Enumeration	Description
Manual	0	Failover must be initiated by the user.
Automatic	1	Failover will occur automatically when a failover condition is met.

FileSystemAttributes

Parameter of

LogicalVolume on page 280, PhysicalItem on page 299, PhysicalVolume on page 302, Volume on page 353, VolumeOptions on page 356

Properties

Name	Enumeration	Description
ReadOnly	1	The file system item is read-only.
Hidden	2	The file system item is hidden.
System	4	The file system item is a system item.
Directory	16	The file system item is a directory.
Archive	32	The file system item is is marked to be archived.
Normal	128	The file system item is normal.
Temporary	256	The file system item is temporary.
SparseFile	512	The file system item is a sparse file.
ReparsePoint	1024	The file system item is a reparse point.
Compressed	2048	The file system item is compressed.
Offline	4096	The file system item is offline.
NotContentIndexed	8192	The file system item is not content indexed.
Encrypted	16384	The file system item is encrypted.

Health

Parameter of

ExtendedLowLevelStates on page 255, JobStatus on page 277

Properties

Name	Enumeration	Description
Unknown	0	The health of the job is unknown or not yet determined.
OK	1	The health of the job is good and data is protected.
Warning	2	The job is taking corrective action. Data may not be protected. You should closely monitor a job in a warning state.
Error	3	The job has encountered an error that requires user attention. Data is not protected.

HighAvailabilityState

Parameter of

CoreMonitorDetails on page 236

Properties

Name	Enumeration	Description
Illegal	-1	There is a problem determining the monitoring status.
None	0	The monitoring status has not yet been determined.
FailoverMonitoring	16	The target is monitoring the source for a failure. No failover condition has been met.
FailoverRequired	32	Failover is pending because a failover condition has been met but failover has not started yet.
FailoverOccurring	48	Failover is occurring.
FailbackRequired	64	The source is failed over to the target. A failback needs to occur to restore the target to its original identity.
FailbackOccurring	80	Failback is occurring.
FailbackRemonitor	96	Failback is complete. The target is waiting to start monitoring the source again.

HighLevelState

Parameter of

JobStatus on page 277

Properties

Name	Enumeration	Description
Unknown	0	The job's state is unknown or not yet determined.
Created	1	The job is being created.
Deleting	2	The job is being deleted.
FailedBack	3	The job has failed back.
FailedOver	4	The job has failed over.
FailingBack	5	The job is failing back.
FailingOver	6	The job is failing over.
FailoverFailed	7	Failover has failed.
FailoverPending	8	The job has a failover condition met.
Mirroring	9	The job is mirroring.
MirrorRequired	10	The job requires a mirror to ensure data integrity.
Paused	11	The job is paused.
Pausing	12	The job is pausing.
Protecting	13	The job is started and protecting.
Provisioning	14	The job is provisioning resources required to start protection.

Name	Enumeration	Description
Restored	15	The job has completed restoring data.
RestoreFailed	16	Restoration has failed.
RestorePaused	17	The job is restoring data but transmission is currently paused.
RestoreRequired	18	The job needs to restore data from the target to a new or the original source.
Restoring	19	The job is restoring data.
Resuming	20	The job is resuming.
Reversing	21	The job is reversing roles.
Reverting	22	The job is reverting.
Starting	23	The job is starting.
Stopped	24	The job is stopped.
Stopping	25	The job is stopping.
Undoing	26	The job is undoing a failover.
RevertingSnapshot	27	The job is reverting to a previous snapshot.
UpdatingTargetImage	28	This property is no longer used.
CredentialsRequired	29	The job does not have sufficient credentials to operate properly.
ActivationCodeWarning	30	The Carbonite replication engine is reporting a licensing warning.
ActivationCodeError	31	The Carbonite replication engine is reporting a licensing error.
EngineConnectionWarning	32	The connection associated with the job is reporting an error, but the threshold for failed communication has not yet been met.

Name	Enumeration	Description
EngineConnectionError	33	The connection associated with the job is reporting an error, and the threshold for failed communication has been exceeded.
EngineServiceWarning	34	The Management Service on the target cannot communicate with the target replication engine or the Management Service on the source cannot communicate with the source replication engine, but the threshold for failed communication has not yet been met.
EngineServiceError	35	The Management Service on the target cannot communicate with the target replication engine or the Management Service on the source cannot communicate with the source replication engine, and the threshold for failed communication has been exceeded.
ServerCommunicationWarning	36	The Management Service on the target cannot contact the Management Service on the source, but the threshold for failed communication has not yet been met.
ServerCommunicationError	37	The Management Service on the target cannot contact the Management Service on the source, and the threshold for failed communication has not yet been met.
TargetInfoNotAvailable	38	The controller appliance for an agentless vSphere job cannot communicate with the target.
Editing	39	The job is being edited.

InclusionMode

Parameter of

PhysicalRule on page 301

Properties

Name	Enumeration	Description
Include	0	The data path is included in replication.
Exclude	1	The data path is excluded from replication.

LicenseType

Parameter of

ActivationCode on page 208

Properties

Name	Enumeration	Description
NotApplicable	0	The license type is not applicable.
Limited	1	The license type is an evaluation license.
Single	2	The license type is a single server license.
Site	3	The license type is a site license.
NodeLockedKey	4	The key is an activation key.

MirrorComparisonCriteria

Parameter of

MirrorParameters on page 286, RestoreParameters on page 317, VerifySchedule on page 347

Properties

Name	Enumeration	Description
None	0	Carbonite will not perform any comparison between the files on the source and target.
Newer	1	Carbonite will compare file attributes between the files on the source and target.
Checksum	2	Carbonite will compare file attributes and file data between files on the source and target.

MirrorOperationOptions

Parameter of

MirrorParameters on page 286, VerifySchedule on page 347

Properties

Name	Enumeration	Description
None	0	Carbonite will not mirror any files.
Synchronize	1	Carbonite will mirror the files from the source to the target. If you use this property by itself, all files will be mirror. If you use this property with CalculateDifferences, only the differences will be mirrored.
Report	2	Carbonite will only report the differences found between the source protected data set and the replica on the target. You must use this option with CalculateDifferences in order for the differences to be reported.
CalculateDifferences	4	The mirroring operation will calculate the differences between the source protected data and the replica on the target using MirrorComparisonCriteria on page 402
CalculateSize	8	The mirroring operation will calculate the size of the source protected data set.
ProcessOrphans	16	The mirroring operation will process orphan files (files in the target path location that are not present on the source).

MirrorState

Returned by

Wait-DtMirrorComplete on page 201

Parameter of

CoreConnectionDetails on page 231

Properties

Name	Enumeration	Description
Calculating	0	The size of the replication set is being calculated.
Idle	1	No data is being mirrored to the target.
Mirror	2	Data is being mirrored to the target.
Pause	3	Mirroring is paused.
RemoveOrphans	4	Orphan files (files in the target path location that are not present on the source) are being deleted.
RepsetVerify	5	The source protected data set is being verified against the target replica data.
Restore	6	Replica data from the target is being restored to the source.
Stopped	7	Mirroring is stopped.
Waiting	8	Mirroring is complete, but data is still being written to the target.
Unknown	9	The mirror is in an unknown or error state.

OperatingSystemArchitecture

Parameter of

OperatingSystemInfo on page 293

Properties

Name	Enumeration	Description
x86	0	The operating system uses the 32-bit architecture.
ia64	6	The operating system uses the Itanium 64-bit architecture.
x64	9	The operating system uses the AMD 64-bit architecture.

OperatingSystemProductType

Parameter of

OperatingSystemInfo on page 293

Properties

Name	Enumeration	Description
None	0	The operating system product type is not defined.
Workstation	1	The operating system is a workstation.
DomainController	2	The operating system is a domain controller.
Server	3	The operating system is a server.

PathBlockingMode

Parameter of

PathBlocking on page 297

Properties

Name	Enumeration	Description
Blocked	0	The path is blocked for writing, except by Carbonite.
Unblocked	1	The path is unblocked for writing.

PingMethods

Parameter of

MonitoredAddressConfiguration on page 289

Properties

Name	Enumeration	Description
None	0	The IP address will not be monitored for failure and cannot be failed over.
Network	1	The IP address will be monitored using an ICMP ping.
Service	2	The IP address will be monitored using the Double-Take service.
Manual	4	The IP address will not be monitored for failure but is still eligible for failover.
Script	8	The IP address will be monitored using a user scripted ping method. If you use this value, you must set MonitorConfiguration on page 287.ScriptMonitorEngine and MonitorConfiguration on page 287.ScriptMonitorName.
ForceUpdate	256	Do not use this property. Carbonite uses it internally.

RecursionMode

Parameter of

PhysicalRule on page 301

Properties

Name	Enumeration	Description
Recursive	0	The physical rule will be applied to this path and all of its children paths.
NonRecursive	1	The physical rule will be applied to this path only.

ReplicationSetUsageType

Parameter of

TargetStateInfo on page 337

Properties

Name	Enumeration	Description
Invalid	-1	This value is an unknown data set type.
Normal	0	This value is for all job types.
SystemState	1	This property is no longer used.
GeoCluster	2	This property is no longer used.

ReplicationState

Parameter of

CoreConnectionDetails on page 231

Properties

Name	Enumeration	Description
NotReplicating	0	Replication is stopped.
OutOfMemory	1	Replication memory has been exhausted.
Pending	2	Replication is pending.
Replicating	3	Data is being replicated to the target.
Unknown	4	Replicationis in an unknown or error state.
Watchdog	5	The Double-Take service is not receiving replication operations from the Carbonite driver. Check the event log for driver related issues
Ready	6	There is no data to replicate.

RestoreParametersRestoreOptions

Parameter of

RestoreParameters on page 317

Properties

Name	Enumeration	Description
None	0	No additional restroration options will be used.
UseTargetWorkload	1	Use the workload information that is stored on the target instead of any workload information that is persisted on the source.
RestoreWorkloadToSource	2	Restore the workload information from the target to the source. This property can only be used with UseTargetWorkload.
OverwriteExistingFiles	4	Overwrite all existing files on the source during the restoration.

RestoreStates

Parameter of

CoreMonitorDetails on page 236

Properties

Name	Enumeration	Description
None	0	There is no restoration connection.
OldServer	1	The server is running an old version of Carbonite and the restoration status is not available.
Required	2	A restoration is required.
Connected	4	The restoration connection is active and is replicating.
MultiConnect	8	There is more than one active restoration connection.
Mirroring	16	The restoration connection is active and mirroring.
MirrorStopped	32	The restoration mirror has been stopped.
OpDropped	64	A mirroring or replication operation has been dropped. A restoration remirorr is required.
OpRetrying	128	A mirorring or replication write operation is being retried.

RestoreStatus

Parameter of

RecommendedFailbackOptions on page 309

Properties

Name	Enumeration	Description
NotStarted	0	The restoration operation has not started.
Restoring	1	The restoration process is in progress.
Restored	2	The restoration process is complete. It is now safe to perform failback.

SaturationLevel

Parameter of

LogicalItems on page 279, LogicalVolume on page 280, PhysicalItem on page 299, PhysicalVolume on page 302, Volume on page 353

Properties

Name	Enumeration	Description
Unknown	0	The saturation level is unknown.
None	1	The item will not be replicated.
Partial	2	Some, but not all, of the item will be replicated.
Full	3	The item and all of its children (if a container) will be replicated.
Error	4	An error occurred while calculating the saturation for the item.

ScriptExecutionMode

Parameter of

ScriptPoint on page 319

Properties

Name	Enumeration	Description
Synchronous	0	Carbonite will wait while the script is executed.
Asynchronous	1	Carbonite will not wait while the script is executed.

ScriptPointType

Parameter of

ScriptPoint on page 319

Properties

Name	Enumeration	Description
MirrorStart	0	The script is executed when the target receives the first mirror operation. In the case of a difference mirror, this may be a long time after the mirror is started because the script does not start until the first different data is received on the target. If the data is synchronized and a difference mirror finds nothing to mirror, the script will not be executed.
MirrorComplete	1	The script is executed when a mirror is completed. Because the mirror statistics may indicate a mirror is at 99-100% when it is actually still processing (for example, if files were added after the job size was calculated, if there are alternate data streams, and so on), the script will not start until all of the mirror data has been completely processed on the target.
MirrorStop	2	The script is executed when a mirror is stopped, which may be caused by an auto-disconnect occurring while a mirror is running, the service is shutdown while a mirror is running, or if you stop a mirror manually.

SmtpConnectionSecurity

Parameter of

EmailNotificationOptions on page 249

Properties

Name	Enumeration	Description
Plain	0	The security is plain.
SSL	1	The security is Secure Sockets Layer.
TLS	2	The security is Transport Layer Security.

SnapshotAttributes

Parameter of

SnapshotEntry on page 329

Properties

Name	Enumeration	Description
None	0	This value represents no snapshots available.
JobManaged	1	This value represents a snapshot managed by a job.
ActiveTest	2	This value represents a snapshot managed by a test failover.

SnapshotCreationReason

Parameter of

SnapshotEntry on page 329, SourceQueueSnapshotEntry on page 331

Properties

Name	Enumeration	Description
Manual	0	A user manually took this snapshot.
Automatic	1	Carbonite automatically took this snapshot.
Scheduled	2	A periodic snapshot schedule triggered this snapshot.
Deferred	3	A periodic snapshot schedule triggered this snapshot, although it did not occur at the specified interval because the job between the source and target was not in a good state
DataTest	4	The test failover process took this snapshot.
Coordinated	5	A user took a coordinated snapshot.
SQLClusterAutomatic	6	The test failover process took this snapshot for a clustered SQL job.

SnapshotQuality

Parameter of

SourceQueueSnapshotEntry on page 331

Properties

Name	Enumeration	Description
Good	0	This value represents a good snapshot.
Bad	1	This value represents a bad snapshot.

SnapshotState

Parameter of

SourceQueueSnapshotEntry on page 331

Properties

Name	Enumeration	Description
Completed	0	This value represents a completed snapshot.
Pending	1	This value represents a pending snapshot.
Error	2	This value represents a snapshot error.

TargetServiceStatus

Parameter of

TargetServicesToStop on page 336

Properties

Name	Enumeration	Description
Stopped	0	The service is stopped.
Started	1	The service is started.
StopPending	2	The service is pending a stop.
StartPending	3	The service is pending a start.
ResumePending	4	The service is pending a resume.
PausePending	5	The service is pending a pause.
Paused	6	The service is paused.
Unknown	7	The state of the service is unknown.

TargetStates

Parameter of

CoreConnectionDetails on page 231, SnapshotEntry on page 329, SourceQueueSnapshotEntry on page 331, TargetStateInfo on page 337

Properties

Name	Enumeration	Description
Good	0	The target is in a good state.
Mirroring	1	Mirroring is in progress.
MirrorStopped	2	Mirroring was stopped. A remirror should be performed.
OpDropped	4	The target detected than an operation was dropped on the network. A remirror should be performed.
Retrying	8	The target is retrying a write operation.
Paused	16	Writing to the target has been paused by the user.
PausePending	32	Writing to the target has been paused by the user.
RestoreRequired	64	The data on the source and target may not be synchronized because of a failover condition. This state will remain until a restore or remirror is completed.
ReplicationPending	128	The connection has been established and replication is enabled, but the first replication operation has not been transmitted yet
SnapshotReverted	256	The data on the source and target may not be synchronized because a snapshot was applied on the target. This state will remain until a restore or remirror is completed.
FailoverUnblocked	512	This property is no longer used.

Name	Enumeration	Description
Disconnected	1024	The target server cannot be contacted.
SroImage	2048	This property is no longer used.
FailoverMonitoring	4096	The target is monitoring the source for a failure.
TransactionsPending	8192	There are transactional NTFS (TxF) operations pending.
GCReplicationComplete	16384	This property is no longer used.
MarkedForDeletion	32768	The connection is marked for deletion when the source comes back online.
JobNotReadyOnTarget	65536	The target server may not be ready to resume transmission, for example due to disks being offline.
TargetPathBlocked	2147483648	Writing to the replica path location on the target is blocked.
Unknown	-4294967296	The target state cannot be determined.

TransmissionMode

Parameter of

CoreConnectionDetails on page 231

Properties

Name	Enumeration	Description
Error	0	Transmission is in an error state.
Paused	1	Transmission is paused.
Started	2	Transmission is active.
Scheduled	3	Transmission is pending a schedule.
Stopped	4	Transmission is stopped.
Unknown	5	Transmission is unknown.

VmwareCertificatePolicy

Parameter of

Set-DtVmwareCertificatePolicy on page 151

Properties

Name	Enumeration	Description
AllowAll	0	Allow all certificates to be installed.
AllowSelfSigned	1	Allow only self-signed certificates to be installed.
AllowValid	2	Allow only valid certificates to be installed.
AllowKnownOrValid	4	Allow only known or valid certificates to be installed.

Weekdays

Parameter of

BandwidthEntry on page 216, BandwidthScheduleEntry on page 220

Properties

Name	Enumeration	Description
None	0	No days of the week are specified.
Sunday	1	Sunday is specified.
Monday	2	Monday is specified.
Tuesday	4	Tuesday is specified.
Wednesday	8	Wednesday is specified.
Thursday	16	Thursday is specified.
Friday	32	Friday is specified.
Workdays	62	Monday, Tuesday, Wednesday, Thursday, and Friday are specified.
Saturday	64	Saturday is specified.
Weekends	65	Saturday and Sunday are specified.
All	127	All days of the week are specified.

Chapter 5 Scripting examples

Below are links to sample Carbonite PowerShell scripts. The sample scripts must be modified. They cannot be used as-is. Modify them to fit your environment. If you need basic assistance with script modifications, contact Technical Support. Assistance with advanced scripting will be referred to Professional Services.

Job creation scripts

Below are links to sample job creation scripts. The sample scripts must be modified. They cannot be used as-is. Modify them to fit your environment. If you need basic assistance with script modifications, contact Technical Support. Assistance with advanced scripting will be referred to Professional Services.

- Carbonite Availability
  - Creating a files and folders job for Windows on page 431
  - Creating a full server job for Windows on page 433
  - Creating a full server job for Linux on page 435
  - Creating a SQL job on page 437
  - Creating a full server to ESX job for Windows on page 439
  - Creating a full server to ESX job for Linux on page 442
  - Creating a full server to Hyper-V job on page 446
- Carbonite Migrate
  - Creating a files and folders migration job for Windows on page 448
  - Creating a full server migration job for Windows on page 450
  - Creating a full server to ESX migration job for Windows on page 452
  - Creating a full server to Hyper-V migration job on page 454

Creating a files and folders job for Windows

The following sample script will create a simple files and folders job for Windows. You will need to modify this script to fit your environment and configuration. If your source or target is a cluster, additional parameters must be added.

The following lines are wrapped to the line below so that you can see all of the text on the page.

- $DtProtectionPath1 =
- $DtProtectionRule1 =
- $DtProtectionPath2 =
- $DtProtectionRule2 =
- $DtJobOptions =
- $DtJobGuidForFilesAndFolders =
- # $DtJobGuidForFilesAndFolders =

When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to create a simple files and folders job

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

     # Type of workload you will be protecting and type of job you will be creating
     $DtWorkloadType = "FilesAndFolders"
     $DtJobType = "FilesAndFolders"

     # Paths on the source to protect
     $DtSourceProtectionPath1 = "C:\Dir1\"
     $DtSourceProtectionPath2 = "C:\Dir2\"

     # Path mapping that will be used when the job is created
     $DtSourcePath = "C:\"
     $DtTargetPath = "C:\Dir3\"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Create a workload
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -WorkloadTypeName $DtWorkloadType

# Specify the files and folders to protect
$DtProtectionPath1 = New-Object DoubleTake.Common.Contract.PhysicalRule -Property @
{Path=$DtSourceProtectionPath1}
$DtProtectionRule1 = Add-DtPhysicalRule -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -Rule
$DtProtectionPath1
$DtProtectionPath2 = New-Object DoubleTake.Common.Contract.PhysicalRule -Property @
```

```
{Path=$DtSourceProtectionPath2}
$DtProtectionRule2 = Add-DtPhysicalRule -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -Rule
$DtProtectionPath2
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid

# Get the default job options that will be used to create the job
$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType
-Workload $DtWorkload

# Sets the path mapping on the target to an all-to-one location
$DtJobOptions.JobOptions.CoreConnectionOptions.PathTransformations[0].SourcePath = $DtSourcePath
$DtJobOptions.JobOptions.CoreConnectionOptions.PathTransformations[0].TargetPath = $DtTargetPath

# Create the job
$DtJobGuidForFilesAndFolders = New-DtJob -ServiceHost $DtTarget -Source $DtSource -JobType
FilesAndFolders -JobOptions $DtJobOptions.JobOptions

# If you do not want to specify job options and instead use the default options,
# remove the PathTransformations lines above and use the New-DtFilesAndFoldersJob
# cmdlet, similar to the following line.
# $DtJobGuidForFilesAndFolders = New-DtFilesAndFoldersJob -ServiceHost $DtTarget -Source $DtSource -Path
$DtSourcePath -JobOptions $DtJobOptions.JobOptions

# Start the job
Start-DtJob -ServiceHost $DtTarget -JobId $DtJobGuidForFilesAndFolders

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Creating a full server job for Windows

The following sample script will create a simple full server job for Windows . You will need to modify this script to fit your environment and configuration.

The following lines are wrapped to the line below so that you can see all of the text on the page.

- # Set-DtLogicalItemSelection
- $DtJobOptions =
- $DtJobGuidForFullServer =
- $TestServer =
- $DtJobOptions.JobOptions.FullServerTestFailoverOptions.TestFailoverServerCredential

When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to create a simple full server for Windows job

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

     # Reserved IP addresses of source and target
     $DtSourceReservedIP = "10.10.10.29"
     $DtTargetReservedIP = "10.10.10.30"

     # Type of workload you will be protecting and type of job you will be creating
     $DtWorkloadType = "FullServerFailover"
     $DtJobType = "FullServerFailover"

     # Test failover server
     $DtTestServerIP = "112.42.75.60"
     $DtTestServerUserName = "domain\administrator"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Create a workload
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -WorkloadTypeName $DtWorkloadType

# This workload, by default, selects all volumes for protection
# If desired, exclude any volumes from protection, however, be careful
# when excluding data as it may compromise the integrity of your installed applications
# Uncomment and use the following line, substituting G:\for the volume you want to exclude
# Repeat the line to exclude multiple volumes
# Set-DtLogicalItemSelection -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -LogicalPath "G:\" -
Unselect

# Get the workload definition including the workload and logical items
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID

# Get the default options that will be used to create the job
```

```
$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType
-Workload $DtWorkload

# Specify the reserved addresses set earlier to be used in the job options to be used for reverse
$DtJobOptions.JobOptions.SystemStateOptions.SourceReservedAddress = $DtSourceReservedIP
$DtJobOptions.JobOptions.SystemStateOptions.TargetReservedAddress = $DtTargetReservedIP

# If you want to disable reverse, you would not need the two lines above.
# Instead, use the following line to disable reverse.
# $DtJobOptions.JobOptions.FullServerFailoverOptions.CreateBackupConnection = $false

# Configure a third server for test failover
$DtCredentialEncrypted = Get-Credential $DtTestServerUserName
$TestServer = New-DtUri -NetworkId $DtTestServerIP -Credential $DtCredentialEncrypted -Scheme
"SchemeName"
$DtJobOptions.JobOptions.FullServerTestFailoverOptions.TestFailoverServerCredential.TestFailoverServerHos
tUri = $TestServer
$DtJobOptions.JobOptions.FullServerTestFailoverOptions.DeleteSnapshots = $true
$DtJobOptions.JobOptions.FullServerTestFailoverOptions.TestFailoverServerAddress = $DtTestServerIP

# Create the job
$DtJobGuidForFullServer = New-DtJob -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType -Options
$DtJobOptions.JobOptions

# Start the job
Start-DtJob -ServiceHost $DtTarget -JobId $DtjobGuidForFullServer

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Creating a full server job for Linux

The following sample script will create a simple full server job for Linux. You will need to modify this script to fit your environment and configuration.

The following lines are wrapped to the line below so that you can see all of the text on the page.

- # Set-DtLogicalItemSelection
- $DtJobOptions =
- $DtJobGuidForLFFO =
- $TestServer =
- $DtJobOptions.JobOptions.FullServerTestFailoverOptions.TestFailoverServerCredential

When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to create a simple full server for Linux job

# Specify the variables to be used in the script

     # Source server and credentials
     # Be sure to include the communication port with the source server
     $DtSourceName = "112.42.7.63:6325"
     $DtSourceUserName = "root"
     $DtSourcePassword = "password"

     # Target server and credentials
     # Be sure to include the communication port with the target server
     $DtTargetName = "112.42.9.93:6325"
     $DtTargetUserName = "root"
     $DtTargetPassword = "password"

     # Reserved IP addresses
     $DtSourceReservedIP = "10.10.10.29"
     $DtTargetReservedIP = "10.10.10.30"

     # Type of workload you will be protecting and type of job you will be creating
     $DtWorkloadType = "LinuxFullServerFailover"
     $DtJobType = "LinuxFullServerFailover"

     # Test failover server
     $DtTestServerIP = "112.42.75.60"
     $DtTestServerUserName = "domain\administrator"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Create a workload
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -WorkloadTypeName $DtWorkloadType

# This workload, by default, selects all volumes for protection
# If desired, exclude any volumes from protection, however, be careful
# when excluding data as it may compromise the integrity of your installed applications
# Uncomment and use the following line, substituting G:\for the volume you want to exclude
# Repeat the line to exclude multiple volumes
# Set-DtLogicalItemSelection -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -LogicalPath "G:\" -
Unselect

# Get the workload definition including the workload and logical items
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID
```

```
# Get the default options that will be used to create the job
$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType
-Workload $DtWorkload

# Specify the reserved addresses set earlier to be used in the job options to be used for reverse
$DtJobOptions.JobOptions.SystemStateOptions.SourceReservedAddress = $DtSourceReservedIP
$DtJobOptions.JobOptions.SystemStateOptions.TargetReservedAddress = $DtTargetReservedIP

# If you want to disable reverse, you would not need the two lines above.
# Instead, use the following line to disable reverse.
# $DtJobOptions.JobOptions.FullServerFailoverOptions.CreateBackupConnection = $false

# Configure a third server for test failover
$DtCredentialEncrypted = Get-Credential $DtTestServerUserName
$TestServer = New-DtUri -NetworkId $DtTestServerIP -Credential $DtCredentialEncrypted -Scheme
"SchemeName"
$DtJobOptions.JobOptions.FullServerTestFailoverOptions.TestFailoverServerCredential.TestFailoverServerHos
tUri = $TestServer
$DtJobOptions.JobOptions.FullServerTestFailoverOptions.DeleteSnapshots = $true
$DtJobOptions.JobOptions.FullServerTestFailoverOptions.TestFailoverServerAddress = $DtTestServerIP

# Create the job
$DtJobGuidForLFFO = New-DtJob -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType -Options
$DtJobOptions.JobOptions

# Start the job
Start-DtJob -ServiceHost $DtTarget -JobId $DtjobGuidForLFFO

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Creating a SQL job

The following sample script will create a simple SQL job. You will need to modify this script to fit your environment and configuration. If your source or target is a cluster, additional parameters must be added.

The following lines are wrapped to the line below so that you can see all of the text on the page.

- $DtProtectionItems =
- # $RootItem =
- # | ForEach-Object
- # $DtProtectionItems
- $DtJobOptions =
- $DtJobGuidForSQL =

When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to create a simple SQL job

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

     # Type of workload you will be protecting and type of job you will be creating
     $DtWorkloadType = "SQL"
     $DtJobType = "SQL"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"


# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Create a workload
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -WorkloadTypeName $DtWorkloadType

# Add what you want to protect to the workload. These lines will, by default, select all instances
$DtLogicalItem = Get-DtLogicalItem -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID
$DtProtectionItems = Set-DtLogicalItemSelection -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -
LogicalPath $DtLogicalItem.Path
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID

# If you do not want to protect all of the instances, as the default does, comment out the
# three lines above and uncomment and use the following group of lines.
# $RootItems = Get-DtLogicalItem -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID
# $RootItems | Format-List
# $RootItem = (Get-DtLogicalItem -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID | Where-Object
{$_.Path -eq "SQL:\"})[ 0 ]
# The following lines unselect all of the instances that were selected by default
# Get-DtLogicalItem -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID -RefItem $RootItem `
#    | ForEach-Object {Set-DtLogicalItemSelection -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID -
LogicalPath $_.Path -Unselect}
# Select the instance that you want to protect by replacing instance_name with the name of the instance
```

```
# For example, SQL:\instance_name would be SQL:\PROD for an instance called PROD
# $DtProtectionItems = Set-DtLogicalItemSelection -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -
LogicalPath "SQL:\instance_name"
# $DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID

# Get the default job options that will be used to create the job
$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType
-Workload $DtWorkload

# Create the job
$DtJobGuidForSQL = New-DtJob -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType -Options
$DtJobOptions.JobOptions

# Start the job
Start-DtJob -ServiceHost $DtTarget -JobId $DtJobGuidForSQL

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Creating a full server to ESX job for Windows

The following sample script will create a full server to ESX job for Windows. It includes options for configuring test failover and reverse. You will need to modify this script to fit your environment and configuration.

The following lines are wrapped to the line below so that you can see all of the text on the page.

- $DtReverseAppliance=
- $VimTarget =
- # Set-DtLogicalItemSelection
- $DtJobOptions =
- The comments in the Configure test failover section
- $DTJobOptions.JobOptions.VRAOptions.SourceApplianceInfo
- $DtJobGuidForEVRA =

When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to create a full server to ESX job
# Including options for test failover and reverse

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

     # If you are configuring a reverse job
     # Reverse target appliance and credentials
     # This may or may not be the same as the target for the forward protection
     $DtReverseTargetName = "gamma"
     $DtReverseTargetUri = "dtms://112.42.76.3:6325"
     $DtReverseTargetRoute = "112.42.76.3"
     $DtReverseTargetUserName = "domain\administrator"
     $DtReverseTargetPassword = "password"

     # ESX host and credentials
     # If you are using vCenter, specify your vCenter.
     # Only specify an ESX host if you are using ESX standalone.
     $DtHostName = "112.42.56.14"
     $DtHostUserName = "root"
     $DtHostPassword = "password"

     # If you are configuring a reverse job
     # Reverse ESX host and credentials
     # If you are using vCenter, specify your vCenter.
     # Only specify an ESX host if you are using ESX standalone.
     $DtReverseHostName = "112.42.12.7"
     $DtReverseHostUserName = "root"
     $DtReverseHostPassword = "password"

     # Type of workload you will be protecting and type of job you will be creating
     $DtWorkloadType = "VRA"
     $DtJobType = "VRA"

     # VM display name
     # This name must be unique within your environment and different
```

```
     # from the existing directory location name if you are reusing an existing disk
     $DisplayName = "Name"

     # Datastore location - GUID assigned to the datastore
     # You can find this GUID in your vSphere or VMware web client
     $DatastoreLocation = "/vmfs/volumes/529a027d-b4ba1124-b1c0-614c42bc0717"

     # If you are configure a reverse job, datastore location
     $ReverseDatastoreLocation = "/vmfs/volumes/630b128d-c5ca2235-b2d1-725d53cd1828"

     # Existing disk - Specify only if you want to reuse an existing disk
     $ExistingDisk = "Dir/SubDir/filename.vmdk”

     # Type of disk. Use Dynamic for ESX thin disks, Fixed for ESX thick disks
     # and Flat Disk for ESX flat disks
     $DiskType = "Dynamic"

     # Test failover configuration
     $SourceVSwitch = "Local Area Connection"
     $TargetVSwitch = "Internal Network"

     # Reverse options
     $ReverseDisplayName = "ReverseName"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# If you are configuring a reverse job, create reverse appliance object
$DtReverseAppliance = New-DtServer -Name $DtReverseTargetName -UserName $DtReverseTargetUserName -
Password $DtReverseTargetPassword -Role ReverseHelperRole

# Create ESX host object
# If you are using vCenter, specify your vCenter.
# Only specify an ESX host if you are using ESX standalone.
$VimTarget = New-DtServer -Name $DtHostName -Username $DtHostUserName -Password $DtHostPassword -Role
TargetVimServer

# If you are configuring a reverse job, create reverse ESX host object
$DtReverseHost = New-DtServer -Name $DtReverseHostName -UserName $DtReverseHostUserName -Password
$DtReverseHostPassword -Role ReverseVimServer

# All roles
$OtherServers = @($VimTarget,$DtReverseHost,$DtReverseAppliance)

# Create a workload
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -WorkloadTypeName $DtWorkloadType

# This workload, by default, selects all volumes for protection
# If desired, exclude any volumes from protection, however, be careful
# when excluding data as it may compromise the integrity of your installed applications
# Uncomment and use the following line, substituting G:\for the volume you want to exclude
# Repeat the line to exclude multiple volumes
# Set-DtLogicalItemSelection -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -LogicalPath "G:\" -
Unselect

# Get the workload definition including the workload and logical items
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID

# Get the default job options that will be used to create the job
$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtTarget -Source $DtSource -OtherServers
$OtherServers -JobType $DtJobType -Workload $DtWorkload

# Set the display name in the job options
$DtJobOptions.JobOptions.VraOptions.ReplicaVmInfo.DisplayName = $DisplayName

# Set the location where you want the VM config files to be located
$DtJobOptions.JobOptions.VRAOptions.ReplicaVMInfo.Path = $DatastoreLocation

# Set the location where you want the VM disk files to be located
foreach($disk in $DtJobOptions.JobOptions.VRAOptions.Volumes)
{
```

```
          $disk.VirtualDiskPath = $DatastoreLocation
          $disk.DiskProvisioningType = $DiskType
          # If you want to reuse an existing disk, you will need to identify the path
          # to use for each $disk using PreexistingDiskPath
}

# Configure test failover
$DTJobOptions.JobOptions.VRAOptions.TestFailover.DeleteVirtualDisks=$true
# If you want to connect the replica to the network, uncomment and use the following lines
# Without these lines, the test failover will be completed without network connectivity
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover = New-Object -TypeName
'DoubleTake.Virtualization.Contract.VirtualSwitchMapping'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[0].SourceVirtualSwitch = New-
Object -TypeName 'DoubleTake.Virtualization.Contract.VirtualSwitchInfo'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[0].TargetVirtualSwitch = New-
Object -TypeName 'DoubleTake.Virtualization.Contract.VirtualSwitchInfo'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[0].SourceVirtualSwitch.Label =
"Name of source adapter goes here"
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[0].TargetVirtualSwitch.Label =
"Name of target virtual switch goes here"
# If you have more than one adapter, you need to add the additional adapters to the
VirtualSwitchMappingTestFailover array
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover += New-Object -TypeName
'DoubleTake.Virtualization.Contract.VirtualSwitchMapping'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[1].TargetVirtualSwitch = New-
Object -TypeName 'DoubleTake.Virtualization.Contract.VirtualSwitchInfo'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[1].SourceVirtualSwitch = New-
Object -TypeName 'DoubleTake.Virtualization.Contract.VirtualSwitchInfo'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[1].SourceVirtualSwitch.Label =
"Name of another source adapter goes here"
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[1].TargetVirtualSwitch.Label =
"Name of target virtual switch goes here"

# Configure reverse job options
# Identify the reverse host
$DTJobOptions.JobOptions.VRAOptions.ReverseOptions.ReverseVMwareServer = $DtReverseHost.Uri
# Identify the reverse appliance by creating a new VMInfo object
$DTJobOptions.JobOptions.VRAOptions.SourceApplianceInfo = New-Object -TypeName
'Doubletake.Virtualization.Contract.VMInfo'
$DTJobOptions.JobOptions.VRAOptions.SourceApplianceInfo.Address = $DtReverseTargetName
$DTJobOptions.JobOptions.VRAOptions.SourceApplianceInfo.GuestUri = $DtReverseTargetURI
# Identify the reverse route
$DTJobOptions.JobOptions.VRAOptions.ReverseRoute = $DtReverseTargetRoute
# Identify the reverse options
$DTJobOptions.JobOptions.VRAOptions.ReverseOptions.VmName = $ReverseDisplayName
$DTJobOptions.JobOptions.VRAOptions.ReverseOptions.VmPath = $ReverseDatastoreLocation
foreach($reverseDisk in $DtJobOptions.JobOptions.VRAOptions.ReverseOptions.Volumes)
{
          $reverseDisk.VirtualDiskPath = $ReverseDatastoreLocation
          $reverseDisk.DiskProvisioningType = $DiskType
          # If you want to reuse an existing disk, you will need to identify the path
          # to use for each $disk using PreexistingDiskPath
}

# Create the job
$DtJobGuidForEVRA = New-DtJob -ServiceHost $DtTarget -Source $DtSource -OtherServers $OtherServers -
JobType $DtJobType -JobOptions $DtJobOptions.JobOptions

# Start the job
Start-DtJob -ServiceHost $DtTarget -JobId $DtJobGuidForEVRA

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
Disconnect-DtServer -ServiceHost $DtTarget
Disconnect-DtServer -ServiceHost $VimTarget
Disconnect-DtServer -ServiceHost $DtReverseAppliance
Disconnect-DtServer -ServiceHost $DtReverseHost
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Creating a full server to ESX job for Linux

The following sample script will create a simple full server to ESX job for Linux. You will need to modify this script to fit your environment and configuration.

The following lines are wrapped to the line below so that you can see all of the text on the page.

- $VimTarget =
- # Set-DtLogicalItemSelection
- $DtJobOptions =
- $ExistingDisk +=
- $ExistingDisk +=
- $ExistingDisk +=
- $DtJobGuidForLVRA =

When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to create a simple full server to ESX job for Linux

# Specify the variables to be used in the script

     # Source server and credentials
     # Be sure to include the communication port with the source server
     $DtSourceName = "112.42.7.63:6325"
     $DtSourceUserName = "root"
     $DtSourcePassword = "password"

     # Target appliance and credentials
     # Be sure to include the communication port with the target
     $DtTargetName = "112.42.9.93:6325"
     $DtTargetUserName = "root"
     $DtTargetPassword = "password"

     # ESX host and credentials
     # If you are using vCenter, specify your vCenter.
     # Only specify an ESX host if you are using ESX standalone.
     $DtHostName = "112.42.56.14"
     $DtHostUserName = "root"
     $DtHostPassword = "password"

     # Type of workload you will be protecting and type of job you will be creating
     $DtWorkloadType = "Lvra"
     $DtJobType = "Lvra"

     # VM display name
     # This name must be unique within your environment and different
     # from the existing directory location name if you are reusing an existing disk
     $DisplayName = "Name"

     # Datastore location - GUID assigned to the datastore
     # You can find this GUID in your vSphere or VMware web client
     $DatastoreLocation = "/vmfs/volumes/529a027d-b4ba1124-b1c0-614c42bc0717"

     # Existing disk - Specify only if you want to reuse an existing disk
     # If your disks are mounted directly (without using LVM2), specify the mount name and location
     # If your disks are in an LVM2 volume group, specify the LVM name and location
     # If you are using both, specify both
     $MountName = "/boot"
     $MountLocation = "Dir1/DiskName__boot.vmdk"
     $LvmName1 = "VG_Name"
     $LvmLocation1 = "Dir1/DiskName_VG_Name_PhysicalVolume0.vmdk"
     $LvmName2 = "VG_Name"
```

```
     $LvmLocation2 = "Dir1/DiskName_VG_Name_PhysicalVolume1.vmdk"

     # Test failover configuration
     $SourceVSwitch = "Local Area Connection"
     $TargetVSwitch = "Internal Network"
     $TestDisplayName = "TestName"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Create ESX host appliance object
# If you are using vCenter, specify your vCenter.
# Only specify an ESX host if you are using ESX standalone.
$VimTarget = New-DtServer -Name $DtHostName -Username $DtHostUserName -Password $DtHostPassword -Role
TargetVimServer
$OtherServers = @($VimTarget)

# Create a workload
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -WorkloadTypeName $DtWorkloadType

# This workload, by default, selects all volumes for protection
# If desired, exclude any volumes from protection, however, be careful
# when excluding data as it may compromise the integrity of your installed applications
# Uncomment and use the following line, substituting G:\for the volume you want to exclude
# Repeat the line to exclude multiple volumes
# Set-DtLogicalItemSelection -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -LogicalPath "G:\" -
Unselect

# Get the workload definition including the workload and logical items
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID

# Get the default job options that will be used to create the job
$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtTarget -Source $DtSource -OtherServers
$OtherServers -JobType $DtJobType -Workload $DtWorkload

# Set the display name
$DtJobOptions.JobOptions.VraOptions.ReplicaVmInfo.DisplayName = $DisplayName

# Set the location where you want the VM config files to be located
$DtJobOptions.JobOptions.VRAOptions.ReplicaVMInfo.Path = $DatastoreLocation

# Uncomment and use the following block only if you want to reuse an existing
# disk with a per volume disk configuration strategy
# Specify the existing disk to use
<#
     $ExistingDisks = @()
     $ExistingDisksDatastoreLocation = $ReplicaVmDatastoreLocation
     $ExistingDisks += @{ mountPoint = $MountName; dsLoc = $ExistingDisksDatastoreLocation; vmdkPath =
$MountLocation }
     $ExistingDisks += @{ vgName = $LvmName1; dsLoc = $ExistingDisksDatastoreLocation; vmdkPath =
$LvmLocation1 }
     $ExistingDisks += @{ vgName = $LvmName2; dsLoc = $ExistingDisksDatastoreLocation; vmdkPath =
$LvmLocation2 }

     $modifiedVolumeGroups = @{}
     foreach($existingDisk in $ExistingDisks)
     {
          if ($existingDisk.ContainsKey("mountPoint"))
          {
               $mntpt = $existingDisk["mountPoint"]
               if ($mntpt)
               {
                    foreach($mountedPartition in $DtJobOptions.JobOptions.VRAOptions.Volumes)
                    {
                         if ($mntpt -eq $mountedPartition.Name)
                        {
                              $mountedPartition.VirtualDiskPath = $existingDisk["dsLoc"]
                              $mountedPartition.PreexistingDiskPath = $existingDisk["vmdkPath"]
                              break
                         }
                    }
```

```
                }
            }
            elseif ($existingDisk.ContainsKey("vgName"))
            {
                # Find the matching VolumeGroup object, clearing any PhysicalVolume instances defined
                # by Get-DtRecommendedJobOptions
                $vgName = $existingDisk["vgName"]
                $matchingVG = $null
                if ($modifiedVolumeGroups.ContainsKey($vgName))
                {
                    $matchingVG = $modifiedVolumeGroups[$vgName]
                }
                else
                {
                    foreach ($vg in $DtJobOptions.JobOptions.VRAOptions.LvmOptions.VolumeGroup)
                    {
                        if ($vg.Name -eq $vgName)
                        {
                            $matchingVG = $vg
                            $matchingVG.PhysicalVolume = @()
                            $modifiedVolumeGroups[$vgName] = $matchingVG
                            break
                        }
                    }
                }
                # Add this existing disk to the VolumeGroup as a PhysicalVolume
                if ($matchingVG)
                {
                    $newPV = New-Object -TypeName DoubleTake.Core.Contract.UVRA.PhysicalVolume
                    $newPV.VirtualDiskPath = $existingDisk["dsLoc"]
                    $newPV.PreexistingDiskPath = $existingDisk["vmdkPath"]
                    $matchingVG.PhysicalVolume += $newPV
                }
                else
                {
                    Write-Error "Volume Groups not found for " + $matchingVG
                }
            }
        }
#>

# Configure test failover
$DTJobOptions.JobOptions.VRAOptions.TestFailover.ReplicaDisplayName = $TestName
$DTJobOptions.JobOptions.VRAOptions.TestFailover.DeleteVirtualDisks=$true
# If you want to connect the replica to the network, uncomment and use the following lines
# Without these lines, the test failover will be completed without network connectivity
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover = New-Object -TypeName
'DoubleTake.Virtualization.Contract.VirtualSwitchMapping'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[0].SourceVirtualSwitch = New-
Object -TypeName 'DoubleTake.Virtualization.Contract.VirtualSwitchInfo'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[0].TargetVirtualSwitch = New-
Object -TypeName 'DoubleTake.Virtualization.Contract.VirtualSwitchInfo'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[0].SourceVirtualSwitch.Label =
"Name of source adapter goes here"
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[0].TargetVirtualSwitch.Label =
"Name of target virtual switch goes here"
# If you have more than one adapter, you need to add the additional adapters to the
VirtualSwitchMappingTestFailover array
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover += New-Object -TypeName
'DoubleTake.Virtualization.Contract.VirtualSwitchMapping'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[1].TargetVirtualSwitch = New-
Object -TypeName 'DoubleTake.Virtualization.Contract.VirtualSwitchInfo'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[1].SourceVirtualSwitch = New-
Object -TypeName 'DoubleTake.Virtualization.Contract.VirtualSwitchInfo'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[1].SourceVirtualSwitch.Label =
"Name of another source adapter goes here"
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[1].TargetVirtualSwitch.Label =
"Name of target virtual switch goes here"

# Create the job
$DtJobGuidForLVRA = New-DtJob -ServiceHost $DtTarget -Source $DtSource -OtherServers $OtherServers -
JobType $DtJobType -JobOptions $DtJobOptions.JobOptions

# Start the job
Start-DtJob -ServiceHost $DtTarget -JobId $DtJobGuidForLVRA
```

```
# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
Disconnect-DtServer -ServiceHost $DtTarget
Disconnect-DtServer -ServiceHost $VimTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Creating a full server to Hyper-V job

The following sample script will create a simple full server to Hyper-V job. You will need to modify this script to fit your environment and configuration. If your source or target is a cluster, additional parameters must be added.

The following lines are wrapped to the line below so that you can see all of the text on the page.

- $DtLogicalItems =
- # Set-DtLogicalItemSelection
- $DtJobOptions =
- $DtJobGuidForHVRA =

When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to create a simple full server to Hyper-V job

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

     # Type of workload you will be protecting and type of job you will be creating
     $DtWorkloadType = "VRA"
     $DtJobType = "VRA"

     # Test failover configuration
     $SourceVSwitch = "Local Area Connection"
     $TargetVSwitch = "Internal Network"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Create a workload
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -WorkloadTypeName $DtWorkloadType

# This workload, by default, selects all volumes for protection
# If desired, exclude any volumes from protection, however, be careful
# when excluding data as it may compromise the integrity of your installed applications
# Uncomment and use the following line, substituting G:\for the volume you want to exclude
# Repeat the line to exclude multiple volumes
# Set-DtLogicalItemSelection -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -LogicalPath "G:\" -
Unselect

# Get the workload definition including the workload and logical items
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID

# Get the default job options that will be used to create the job
$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType
-Workload $DtWorkload

# Configure test failover
$DTJobOptions.JobOptions.VRAOptions.TestFailover.DeleteVirtualDisks=$true
# If you want to connect the replica to the network, uncomment and use the following lines
```

```
# Without these lines, the test failover will be completed without network connectivity
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover = New-Object -TypeName
'DoubleTake.Virtualization.Contract.VirtualSwitchMapping'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[0].SourceVirtualSwitch = New-
Object -TypeName 'DoubleTake.Virtualization.Contract.VirtualSwitchInfo'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[0].TargetVirtualSwitch = New-
Object -TypeName 'DoubleTake.Virtualization.Contract.VirtualSwitchInfo'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[0].SourceVirtualSwitch.Label =
"Name of source adapter goes here"
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[0].TargetVirtualSwitch.Label =
"Name of target virtual switch goes here"
# If you have more than one adapter, you need to add the additional adapters to the
VirtualSwitchMappingTestFailover array
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover += New-Object -TypeName
'DoubleTake.Virtualization.Contract.VirtualSwitchMapping'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[1].TargetVirtualSwitch = New-
Object -TypeName 'DoubleTake.Virtualization.Contract.VirtualSwitchInfo'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[1].SourceVirtualSwitch = New-
Object -TypeName 'DoubleTake.Virtualization.Contract.VirtualSwitchInfo'
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[1].SourceVirtualSwitch.Label =
"Name of another source adapter goes here"
# $DtJobOptions.JobOptions.VRAOptions.VirtualSwitchMappingTestFailover[1].TargetVirtualSwitch.Label =
"Name of target virtual switch goes here"

# Create the job
$DtJobGuidForHVRA = New-DtJob -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType -JobOptions
$DtJobOptions.JobOptions

# Start the job
Start-DtJob -ServiceHost $DtTarget -JobId $DtJobGuidForHVRA

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Creating a files and folders migration job for Windows

The following sample script will create a simple files and folders migration job for Windows. You will need to modify this script to fit your environment and configuration.

The following lines are wrapped to the line below so that you can see all of the text on the page.

- $DtMigrationPath1 =
- $DtMigrationRule1 =
- $DtMigrationPath2 =
- $DtMigrationRule2 =
- $DtJobOptions =
- $DtJobGuidForDataMigration =

When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to create a simple data migration job

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

     # Type of workload you will be migrating and type of job you will be creating
     $DtWorkloadType = "MoveDataOnlyMigration"
     $DtJobType = "MoveDataOnlyMigration"

     # Paths on the source to migrate
     $DtSourceMigrationPath1 = "C:\Dir1\"
     $DtSourceMigrationPath2 = "C:\Dir2\"

     # Path mapping that will be used when the job is created
     $DtSourcePath = "C:\"
     $DtTargetPath = "C:\Dir3\"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Create a workload
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -WorkloadTypeName $DtWorkloadType

# Specify the files and folders to migrate
$DtMigrationPath1 = New-Object DoubleTake.Common.Contract.PhysicalRule -Property @
{Path=$DtSourceMigrationPath1}
$DtMigrationRule1 = Add-DtPhysicalRule -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -Rule
$DtMigrationPath1
$DtMigrationPath2 = New-Object DoubleTake.Common.Contract.PhysicalRule -Property @
{Path=$DtSourceMigrationPath2}
$DtMigrationRule2 = Add-DtPhysicalRule -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -Rule
$DtMigrationPath2
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid
```

```
# Get the default job options that will be used to create the job
$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType
-Workload $DtWorkload

# Sets the path mapping on the target to an all-to-one location
$DtJobOptions.JobOptions.CoreConnectionOptions.PathTransformations[0].SourcePath = $DtSourcePath
$DtJobOptions.JobOptions.CoreConnectionOptions.PathTransformations[0].TargetPath = $DtTargetPath

# Create the job
$DtJobGuidForDataMigration = New-DtJob -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType -
JobOptions $DtJobOptions.JobOptions

# Start the job
Start-DtJob -ServiceHost $DtTarget -JobId $DtJobGuidForDataMigration

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Creating a full server migration job for Windows

The following sample script will create a simple full server migration job for Windows. You will need to modify this script to fit your environment and configuration.

The following lines are wrapped to the line below so that you can see all of the text on the page.

- # Set-DtLogicalItemSelection
- $DtJobOptions =
- $DtJobGuidForFullServerMigration =

When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to create a simple full server migration job

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

     # Type of workload you will be protecting and type of job you will be creating
     $DtWorkloadType = "MoveServerMigration"
     $DtJobType = "MoveServerMigration"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Create a workload
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -WorkloadTypeName $DtWorkloadType

# This workload, by default, selects all volumes for protection
# If desired, exclude any volumes from protection, however, be careful
# when excluding data as it may compromise the integrity of your installed applications
# Uncomment and use the following line, substituting G:\for the volume you want to exclude
# Repeat the line to exclude multiple volumes
# Set-DtLogicalItemSelection -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -LogicalPath "G:\" -
Unselect

# Get the workload definition including the workload and logical items
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID

# Get the default options that will be used to create the job
$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType
-Workload $DtWorkload

# Create the job
$DtJobGuidForFullServerMigration = New-DtJob -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType
-Options $DtJobOptions.JobOptions

# Start the job
Start-DtJob -ServiceHost $DtTarget -JobId $DtJobGuidForFullServerMigration

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
```

```
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Creating a full server to ESX migration job for Windows

The following sample script will create a simple full server to ESX migration job for Windows. You will need to modify this script to fit your environment and configuration.

The following lines are wrapped to the line below so that you can see all of the text on the page.

- $VimTarget =
- # Set-DtLogicalItemSelection
- $DtJobOptions =
- $DtJobGuidForVraMove =

When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to create a simple full server to ESX migration job

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

     # ESX host and credentials
     # If you are using vCenter, specify your vCenter.
     # Only specify an ESX host if you are using ESX standalone.
     $DtHostName = "112.42.56.14"
     $DtHostUserName = "root"
     $DtHostPassword = "password"

    # Type of workload you will be protecting and type of job you will be creating
     $DtWorkloadType = "VraMove"
     $DtJobType = "VraMove"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Create ESX host appliance object
# If you are using vCenter, specify your vCenter.
# Only specify an ESX host if you are using ESX standalone.
$VimTarget = New-DtServer -Name $DtHostName -Username $DtHostUserName -Password $DtHostPassword -Role
TargetVimServer
$OtherServers = @($VimTarget)

# Create a workload
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -WorkloadTypeName $DtWorkloadType

# This workload, by default, selects all volumes for protection
# If desired, exclude any volumes from protection, however, be careful
# when excluding data as it may compromise the integrity of your installed applications
# Uncomment and use the following line, substituting G:\for the volume you want to exclude
# Repeat the line to exclude multiple volumes
# Set-DtLogicalItemSelection -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -LogicalPath "G:\" -
Unselect
```

```
# Get the workload definition including the workload and logical items
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID

# Get the default job options that will be used to create the job
$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType
-Workload $DtWorkload -OtherServers $OtherServers

# Create the job
$DtJobGuidForVraMove = New-DtJob -ServiceHost $DtTarget -Source $DtSource -OtherServers $OtherServers -
JobType $DtJobType -JobOptions $DtJobOptions.JobOptions

# Start the job
Start-DtJob -ServiceHost $DtTarget -JobId $DtJobGuidForVraMove

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
Disconnect-DtServer -ServiceHost $DtTarget
Disconnect-DtServer -ServiceHost $VimTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Creating a full server to Hyper-V migration job

The following sample script will create a simple full server to Hyper-V migration job. You will need to modify this script to fit your environment and configuration.

The following lines are wrapped to the line below so that you can see all of the text on the page.

- # Set-DtLogicalItemSelection
- $DtJobOptions =
- $DtJobGuidForVraMove =

When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to create a simple full server to Hyper-V migration job

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

     # Type of workload you will be protecting and type of job you will be creating
     $DtWorkloadType = "VraMove"
     $DtJobType = "VraMove"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Create a workload
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -WorkloadTypeName $DtWorkloadType

# This workload, by default, selects all volumes for protection
# If desired, exclude any volumes from protection, however, be careful
# when excluding data as it may compromise the integrity of your installed applications
# Uncomment and use the following line, substituting G:\for the volume you want to exclude
# Repeat the line to exclude multiple volumes
# Set-DtLogicalItemSelection -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -LogicalPath "G:\" -
Unselect

# Get the workload definition including the workload and logical items
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGUID

# Get the default job options that will be used to create the job
$DtJobOptions = Get-DtRecommendedJobOptions -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType
-Workload $DtWorkload

# Create the job
$DtJobGuidForVraMove = New-DtJob -ServiceHost $DtTarget -Source $DtSource -JobType $DtJobType -JobOptions
$DtJobOptions.JobOptions

# Start the job
Start-DtJob -ServiceHost $DtTarget -JobId $DtJobGuidForVraMove

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
```

```
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Job information scripts

Below are links to sample job information scripts. The sample scripts must be modified. They cannot be used as-is. Modify them to fit your environment. If you need basic assistance with script modifications, contact Technical Support. Assistance with advanced scripting will be referred to Professional Services.

Viewing job Event messages

Most Carbonite Event messages are located in the Application Log with a Source of Double-Take or Double-Take Management Service. You will also find some Event messages in the System log under RepDrv. See the *Reference Guide* for details on all of the Carbonite Event messages.

The following sample scripts will gather Carbonite specific Event messages. The cmdlets used in these scripts are not Carbonite cmdlets. They are Windows PowerShell cmdlets. See your Windows PowerShell documentation for more details and examples on how to use these cmdlets.

You will need to modify this script to fit your environment and configuration.

Each Get-EventLog cmdlet is just one line. It may be wrapped to the line below so that you can see all of the text on the page. When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to view job Event messages

# Set the date for how far back you want to view
$Date = get-date 01/15/2018

# Display all Double-Take service and Double-Take Management Service Event messages
# since the date you specified
Get-EventLog -LogName Application -Source @("Double-Take", "Double-Take Management Service") -After $Date
```

```
# Sample script to view job Event messages

# Display the last five Double-Take service or Double-Take Management Service Event
# messages, listing all properties of the Events
Get-EventLog -LogName Application -Source @("Double-Take", "Double-Take Management Service") -Newest 5 |
format-list -property *
```

```
# Sample script to view job Event messages

# Set the values of the Event IDs you want to see
$FirstEventId = 4065     # Target data state change
$SecondEventId = 4111    # Sharing violation on target
$ThirdEventId = 8196     # Memory issues on source

# Set the date for how far back you want to view
$Date = get-date 01/15/2018

# Display specific Double-Take service or Double-Take Management Service Event messages
# based on the Event IDs
Get-EventLog -LogName Application -Source @("Double-Take", "Double-Take Management Service") -After $Date
| Where-Object {$_.EventID -eq $FirstEventId -or $_.EventId -eq $SecondEventId}

# Display specific RepDrv Event messages based on the Event IDs
Get-EventLog -LogName System -Source RepDrv -After $Date | Where-Object {$_.EventID -eq $ThirdEventId}
```

```
# Sample script to view job Event messages

# Display specific Double-Take service or Double-Take Management Service Event messages
# based on the Event index number, and listing all properties of the Event
Get-EventLog -LogName Application -Source @("Double-Take", "Double-Take Management Service") | Where-
Object {$_.Index -eq 99461} | format-list -property *
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Creating a job diagnostics file

The following sample script will create a job diagnostics file, also known as DTInfo. The file will be located in the \Service\Data directory where you installed Carbonite. This is a file you may want to give to technical support if you are troubleshooting a job. There will be a separate file for each job on your target. You will need to modify this script to fit your environment and configuration.

```
# Sample script to create a job diagnostics files

# Specify the variables to be used in the script

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Login to your target server
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Get the jobs on the target and pass through to create a diagnostics file
Get-DtJob -ServiceHost $DtTarget | Save-DtJobDiagnostics -ServiceHost $DtTarget

# Close the connections for the server object
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Job control scripts

Below are links to sample job control scripts. The sample scripts must be modified. They cannot be used as-is. Modify them to fit your environment. If you need basic assistance with script modifications, contact Technical Support. Assistance with advanced scripting will be referred to Professional Services.

Validating an existing job

The following sample script will validate an existing job. You will need to modify this script to fit your environment and configuration.

The $DtJob line is wrapped to the line below so that you can see all of the text on the page. When re-creating a script like this for your environment, make sure you enter that command on just one line.

```
# Sample script to validate an existing Carbonite job

# Specify the variables to be used in the script

     # Source server
     $DtSourceName = "alpha"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"


# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"


# Create target object
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Find the appropriate job, based on the source server name.
$DtJob = Get-DtJob -ServiceHost $DtTarget | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DTSourceName}

# Validate the job options.
$DtConfirmation = Confirm-DtJobOptions -ServiceHost $DtTarget -JobId $DtJob.Id -JobOptions $DtJob.Options

# Give the validation process time to complete.
while ($true)
{
          sleep 1
          $DtConfirmStatus = Get-DtVerificationStatus -ServiceHost $DtTarget -Token $DtConfirmation
          if ($DtConfirmStatus.Task.Status -eq "Faulted")
          {
                    throw $("Validation failed: {0}" -f $DtConfirmStatus.Task.MessageId)
          }
          if ($DtConfirmStatus.Task.Status -eq "Completed")
          {
                    break
          }
}
$StatusCount=0
$DtConfirmStatus.Steps | ForEach-Object {
          if ($_.Status -eq "Warning" -or $_.Status -eq "Error")
          {
                    $StatusCount++
                    # For each error or warning, display the level and message.
                     Write-Host "$($_.Status) : $($_.MessageKey)"
          }
}

# Identify if there were no errors or warnings.
if ($StatusCount -eq 0)
{
          Write-Host "No job validation errors or warnings were detected."
}

# Close the connection for the server object
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Editing a files and folders job for Windows

The following sample script will edit an existing files and folders job for Windows. You will need to modify this script to fit your environment and configuration.

The following lines are wrapped to the line below so that you can see all of the text on the page.

- $DtJob =
- $DtNewRule =
- $DtExcludeTxtRule =

When re-creating a script like this for your environment, make sure you enter those commands on just one line.

```
# Sample script to edit an existing files and folders Carbonite job

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

     # Changes to the job
     $DtJobDirectory = "C:\NewDirectory"
     $DtJobFileToExclude = "C:\NewDirectory\*.txt"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Identify the job, based on the source server name
$DtJob = Get-DtJob -ServiceHost $DtTarget | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtSourceName}

# Create a workload object on the source to edit the current workload rules
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -Workload $DtJob.Options.Workload

# Specify the additional files and folders to protect
$DtNewRule = Add-DtPhysicalRule -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -Path $DtJobDirectory

# Specify files to exclude from protection, in this example .txt files in the new protection rule
$DtExcludeTxtRule = Add-DtPhysicalRule -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid -Path
$DtJobFileToExclude -Exclude

# Update the workload rules in the job options with the new modifications
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid
$DtJob.Options.Workload=$DtWorkload

# Update the path mapping of the replicated data on the target based on the current recommendations
$DtTargetPath = Get-DtRecommendedPathTransform -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid
$DtJob.Options.CoreConnectionOptions.PathTransformations = $DtTargetPath

# If you do not want to use the one-to-one path mapping in the default recommended options,
# you can configure the job to use specific locations, similar to the following lines.
# $DtJob.Options.CoreConnectionOptions.PathTransformations[0].SourcePath = "C:\"
# $DtJob.Options.CoreConnectionOptions.PathTransformations[0].TargetPath = "C:\ReplicatedData\"
```

```
# Verify the new job options on the existing job.
$DtConfirmation = Confirm-DtJobOptions -ServiceHost $DtTarget -JobId $DtJob.Id -JobOptions $DtJob.Options
do
{
          # Poll every second for the confirmation status
          Start-Sleep -Seconds 1
          $DtConfirmStatus = Get-DtVerificationStatus -ServiceHost $DtTarget -Token $DtConfirmation
          # When the ActivityCompletionStatus is not InProgress, the confirmation is complete.
}
while ($DtConfirmStatus.Steps.Status -eq 0)

# If the ActivityCompletionStatus is Error, print out the steps reporting an Error.
if ($DtConfirmStatus.Steps.Status -eq 3)
{
          Write-Error "The following job validation errors were detected:"
          $DtConfirmStatus.Steps | ForEach-Object
          {
                    if ($_.Status -eq 3)
                    {
                              Write-Error "$($_.Id) : $($_.TitleKey) : $($_.MessageKey)"
                    }
          }
          # Terminate so the job is not edited with invalid options
          throw "Job validation failure."
}

# Apply new job options with the updated workload rules, forcing a remirror.
Edit-DtJob -ServiceHost $DtTarget -JobId $DtJob.Id -JobOptions $DtJob.Options

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Changing the compression setting for an existing job

The following sample script will change the compression setting for an existing job. You will need to modify this script to fit your environment and configuration.

The $DtJob line is wrapped to the line below so that you can see all of the text on the page. When re-creating a script like this for your environment, make sure you enter that command on just one line.

```
# Sample script to change the compression settings for an existing Carbonite job

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Identify the job, based on the source server name
$DtJob = Get-DtJob -ServiceHost $DtTarget | Where-Object {
$_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtSourceName}

# Create a workload object on the source to edit the current workload rules
$DtWorkloadGUID = New-DtWorkload -ServiceHost $DtSource -Workload $DtJob.Options.Workload

# Enable compression using one of the following combinations
# level = -1 Compression is disabled
# level = 0 and algorithm = 10 Compression is enabled at low level
# level = 1 and algorithm = 21 Compression is enabled at medium level
# level = 2 and algorithm = 31 Compression is enabled at high level
$DtJob.Options.CoreConnectionOptions.ConnectionStartParameters.CompressionLevel.Level=1
$DtJob.Options.CoreConnectionOptions.ConnectionStartParameters.CompressionLevel.Algorithm=21

# Update the workload rules in the job options with the new modifications
$DtWorkload = Get-DtWorkload -ServiceHost $DtSource -WorkloadId $DtWorkloadGuid
$DtJob.Options.Workload=$DtWorkload

# Verify the new job options on the existing job.
$DtConfirmation = Confirm-DtJobOptions -ServiceHost $DtTarget -JobId $DtJob.Id -JobOptions $DtJob.Options
do
{
          # Poll every second for the confirmation status
          Start-Sleep -Seconds 1
          $DtConfirmStatus = Get-DtVerificationStatus -ServiceHost $DtTarget -Token $DtConfirmation
          # When the activity completion status is not InProgress, the confirmation is complete.
} while ($DtConfirmStatus.Steps.Status -eq 0)

# If the completion status is Error, print out the steps reporting an Error
if ($DtConfirmStatus.Steps.Status -eq 3)
{
          Write-Error "The following job validation errors were detected:"
          $DtConfirmStatus.Steps | ForEach-Object
          {
                    if ($_.Status -eq 3)
                    {
                              Write-Error "$($_.Id) : $($_.TitleKey) : $($_.MessageKey)"
                    }
```

```
            }
           # Terminate so the job is not edited with invalid options
           throw "Job validation failure."
}

# Apply new job options with the updated workload rules, forcing a remirror.
Edit-DtJob -ServiceHost $DtTarget -JobId $DtJob.Id -JobOptions $DtJob.Options

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Stopping and starting a job

The following sample scripts stop and start a Carbonite job on your target. You will need to modify these scripts to fit your environment and configuration.

```
# Sample script to stop a Carbonite job on your target

# Specify the variables to be used in the script

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create target object
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Get the job ID of the job running on the target
$DtJob = Get-DtJob -ServiceHost $DtTarget

# Stop the job running on the target
Stop-DtJob -ServiceHost $DtTarget -JobId $DtJob.Id

# Close the connections for the server object
Disconnect-DtServer -ServiceHost $DtTarget
```

```
# Sample script to start a Carbonite job on your target

# Specify the variables to be used in the script

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create target object
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Get the job ID of the job running on the target
$DtJob = Get-DtJob -ServiceHost $DtTarget

# Resume the job running on the target
Start-DtJob -ServiceHost $DtTarget -JobId $DtJob.Id

# Close the connections for the server object
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

If you have multiple jobs on your target, you can use the Windows Where-Object cmdlet to identify a specific job by its source URI, source server name, or by job name. For example, you might use one of the following.

```
$DtJob = Get-DtJob -ServiceHost $DtTarget | Where-Object { $_.SourceHostUri.Host -eq “ServerName” }

$DtJob = Get-DtJob -ServiceHost $DtTarget | Where-Object { $_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtJob = Get-DtJob -ServiceHost $DtTarget | Where-Object { $_.Options.Name -eq “source to target” }
```

See your Windows PowerShell documentation for more details on using the Where-Object command.

Pausing and resuming a job

The following sample scripts pause and resume a Carbonite job on your target. You will need to modify these scripts to fit your environment and configuration.

```
# Sample script to pause a Carbonite job on your target

# Specify the variables to be used in the script

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create target object
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Get the job ID of the job running on the target
$DtJob = Get-DtJob -ServiceHost $DtTarget

# Pause the job running on the target
Suspend-DtJob -ServiceHost $DtTarget -JobId $DtJob.Id

# Close the connections for the server object
Disconnect-DtServer -ServiceHost $DtTarget
```

```
# Sample script to resume a Carbonite job on your target

# Specify the variables to be used in the script

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create target object
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Get the job ID of the job running on the target
$DtJob = Get-DtJob -ServiceHost $DtTarget

# Resume the job running on the target
Resume-DtJob -ServiceHost $DtTarget -JobId $DtJob.Id

# Close the connections for the server object
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

If you have multiple jobs on your target, you can use the Windows Where-Object cmdlet to identify a specific job by its source URI, source server name, or job name. For example, you might use one of the following.

```
$DtJob = Get-DtJob -ServiceHost $DtTarget | Where-Object { $_.SourceHostUri.Host -eq “ServerName” }

$DtJob = Get-DtJob -ServiceHost $DtTarget | Where-Object { $_.Statistics.CoreConnectionDetails.SourceMachineName -eq $DtServerObjectAlpha}

$DtJob = Get-DtJob -ServiceHost $DtTarget | Where-Object { $_.Options.Name -eq “source to target” }
```

See your Windows PowerShell documentation for more details on using the Where-Object command.

Viewing and setting job and server options

The following sample script will gather and set several Carbonite job and server options. You may want to consider running cmdlets like this from the PowerShell command line, rather than a script, so you can see the values returned from the get cmdlets and then make appropriate adjustments for your set cmdlets. You will need to modify this script to fit your environment and configuration. The options used in this script are examples. You can get and set any option. See Server and job settings on page 492 for a complete list of options.

```
# Sample script to gather and set Carbonite job and server options
# You may want to run these cmdlets from the PowerShell command prompt
# so that you can see the values returned for each of the get cmdlets
# and then determine appropriate desired values for each option

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Gather and display several job and server settings
# These options are examples. You can get and set any option.
$DtMaxChecksumBlocksSource = Get-DtOption -ServiceHost $DtSource -Name MaxChecksumBlocks
$DtMaxChecksumBlocksTarget = Get-DtOption -ServiceHost $DtTarget -Name MaxChecksumBlocks
$DtMirrorChunkSizeSource = Get-DtOption -ServiceHost $DtSource -Name MirrorChunkSize
$DtMirrorChunkSizeTarget = Get-DtOption -ServiceHost $DtTarget -Name MirrorChunkSize
$DtCalculateByVolumeSource = Get-DtOption -ServiceHost $DtSource -Name CalculateByVolume
$DtCalculateByVolumeTarget = Get-DtOption -ServiceHost $DtTarget -Name CalculateByVolume
$DtAutoRemirrorSource = Get-DtOption -ServiceHost $DtSource -Name AutoRemirror
$DtAutoRemirrorTarget = Get-DtOption -ServiceHost $DtTarget -Name AutoRemirror
write-output "=================="
write-output "These are the current options and values."
write-output "The source is displayed first, and the target is displayed second."
$DtMaxChecksumBlocksSource
$DtMaxChecksumBlocksTarget
$DtMirrorChunkSizeSource
$DtMirrorChunkSizeTarget
$DtCalculateByVolumeSource
$DtCalculateByVolumeTarget
$DtAutoRemirrorSource
$DtAutoRemirrorTarget

# Store the desired value for each job and server setting
$DtMaxChecksumBlocksDesiredValue = @{MaxChecksumBlocks=64}
$DtMirrorChunkSizeDesiredValue = @{MirrorChunkSize=131072}
$DtCalculateByVolumeDesiredValue = @{CalculateByVolume=1}
$DtAutoRemirrorDesiredValue = @{AutoRemirror=1}

# Set the new values
Set-DtOption -ServiceHost $DtSource -Setting $DtMaxChecksumBlocksDesiredValue
Set-DtOption -ServiceHost $DtTarget -Setting $DtMaxChecksumBlocksDesiredValue
Set-DtOption -ServiceHost $DtSource -Setting $DtMirrorChunkSizeDesiredValue
Set-DtOption -ServiceHost $DtTarget -Setting $DtMirrorChunkSizeDesiredValue
Set-DtOption -ServiceHost $DtSource -Setting $DtCalculateByVolumeDesiredValue
Set-DtOption -ServiceHost $DtTarget -Setting $DtCalculateByVolumeDesiredValue
Set-DtOption -ServiceHost $DtSource -Setting $DtAutoRemirrorDesiredValue
Set-DtOption -ServiceHost $DtTarget -Setting $DtAutoRemirrorDesiredValue

# Regather and display the updated values
```

```
$DtMaxChecksumBlocksSource = Get-DtOption -ServiceHost $DtSource -Name MaxChecksumBlocks
$DtMaxChecksumBlocksTarget = Get-DtOption -ServiceHost $DtTarget -Name MaxChecksumBlocks
$DtMirrorChunkSizeSource = Get-DtOption -ServiceHost $DtSource -Name MirrorChunkSize
$DtMirrorChunkSizeTarget = Get-DtOption -ServiceHost $DtTarget -Name MirrorChunkSize
$DtCalculateByVolumeSource = Get-DtOption -ServiceHost $DtSource -Name CalculateByVolume
$DtCalculateByVolumeTarget = Get-DtOption -ServiceHost $DtTarget -Name CalculateByVolume
$DtAutoRemirrorSource = Get-DtOption -ServiceHost $DtSource -Name AutoRemirror
$DtAutoRemirrorTarget = Get-DtOption -ServiceHost $DtTarget -Name AutoRemirror
write-output " "
write-output "=================="
write-output "These are the updated options and values."
write-output "The source is displayed first, and the target is displayed second."
$DtMaxChecksumBlocksSource
$DtMaxChecksumBlocksTarget
$DtMirrorChunkSizeSource
$DtMirrorChunkSizeTarget
$DtCalculateByVolumeSource
$DtCalculateByVolumeTarget
$DtAutoRemirrorSource
$DtAutoRemirrorTarget

# Close the connections for the server objects. You may want to consider using a finally block.
Disconnect-DtServer -ServiceHost $DtSource
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Other sample scripts

Below are links to other sample scripts. The sample scripts must be modified. They cannot be used as-is. Modify them to fit your environment. If you need basic assistance with script modifications, contact Technical Support. Assistance with advanced scripting will be referred to Professional Services.

Pausing and resuming your target

The following sample scripts pause and resume your Carbonite target. (The server itself is not paused. Only Carbonite processing is paused.) You will need to modify these scripts to fit your environment and configuration.

```
# Sample script to pause the Carbonite target

# Specify the variables to be used in the script

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create target object
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Pause all of the Carbonite jobs on the target
Suspend-DtTarget -ServiceHost $DtTarget -All

# Close the connections for the server object
Disconnect-DtServer -ServiceHost $DtTarget
```

```
# Sample script to resume the Carbonite target

# Specify the variables to be used in the script

     # Target server and credentials
     $DtTargetName = "beta"
     $DtTargetUserName = "domain\administrator"
     $DtTargetPassword = "password"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create target object
$DtTarget = New-DtServer -Name $DtTargetName -UserName $DtTargetUserName -Password $DtTargetPassword

# Resume all of the Carbonite jobs on the target
Resume-DtTarget -ServiceHost $DtTarget -All

# Close the connections for the server object
Disconnect-DtServer -ServiceHost $DtTarget
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Shutting down the Double-Take service on a server

The following sample script will login to a Carbonite server and then shutdown the Double-Take service on that server. You will need to modify this script to fit your environment and configuration.

```
# Sample script to shutdown the Double-Take service on a server

# Specify the variables to be used in the script

     # Source server and credentials
     $DtSourceName = "alpha"
     $DtSourceUserName = "domain\administrator"
     $DtSourcePassword = "password"

# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Create source object
$DtSource = New-DtServer -Name $DtSourceName -UserName $DtSourceUserName -Password $DtSourcePassword

# Shutdown the Double-Take service on the server
Stop-DtReplicationService -ServiceHost $DtSource

# Close the connections for the server object
Disconnect-DtServer -ServiceHost $DtSource
```

If you want to hide your user credentials in your script, use the Windows PowerShell Get-Credential cmdlet. The password will not be visible because Windows stores an encrypted password. See Hiding your password in a PowerShell script on page 476 for basic details on using this cmdlet. See your Windows PowerShell documentation for detailed instructions.

Hiding your password in a PowerShell script

The Carbonite PowerShell cmdlets require a server object, and that server object requires user credentials for the specified server. Many corporate security policies do not allow for user passwords to be typed in plain text, which can make scripting difficult. You can use the credential object returned from the Windows PowerShell Get-Credential cmdlet. This password will not be visible because Windows stores an encrypted password. The following sample script logs in to a Carbonite server using a hidden password. See your Windows PowerShell documentation for more details on creating a credential object with Get-Credential. You will need to modify this script to fit your environment and configuration.

```
# Sample script to login to a Carbonite server using a hidden password

# Specify the variables to be used in the script

     # Source server
     $DtSourceName = "alpha"

     # Target server
     $DtTargetName = "beta"


# Import the Carbonite PowerShell module
# This may be \Service\ or \Console\ depending on your installation
Import-Module "C:\Program Files\Carbonite\Replication\Console\DoubleTake.PowerShell.dll"

# Store user credentials in an encrypted form
$DtCredentialEncrypted = Get-Credential domain\administrator

# At this point, you will be prompted to supply the password
# and the credentials will be stored in an encrypted format

# Create source and target objects
$DtSource = New-DtServer -Name $DtSourceName -Credential $DtCredentialEncrypted
$DtTarget = New-DtServer -Name $DtTargetName -Credential $DtCredentialEncrypted

# If you are incorporating this script into another script,
# be sure and close the connections for the server objects
# at the end of the script using the Disconnect-DtServer
# cmdlet. For example,
# Disconnect-DtServer -ServiceHost $DtSource
# Disconnect-DtServer -ServiceHost $DtTarget
# You may want to consider using a finally block.
```

Chapter 6 Carbonite Replication Console Set Options page to JobOptions class mapping

When using the Carbonite Replication Console to create a job, the **Set Options** page filters the job options in order to display only those that are specific to the type of job you are creating. This filtering also applies to server configurations within a job type. For example, a files and folders job from a standalone source to a standalone target will have different job options displayed in the console than a file and folders job from a clustered source to a clustered target.

There is no such filtering available in the JobOptions class because the class must include all possible configurations for all possible job types. This makes the JobOptions class large and potentially confusing.

The sections below correspond to the accordion sections available on the **Set Options** page in the console. These sections will help you identify a JobOptions class for a particular job option. First, expand the section below that corresponds to the console accordion that has the option you want to set. Then, locate the field name from the console accordion in the left column of the table and the corresponding JobOptions class to use in the right column of the table. Keep in mind these caveats about console options compared to JobOptions classes.

- Some console options are for display purposes only and do not have a corresponding JobOptions class.
- Some console options may have more than one JobOptions class that may need to be set.
- Some console options may have unique JobOptions classes depending on the job type.
- The sections below and the information within them are a superset of all available accordions and all available options within an accordion. You may not be able to configure a JobOptions class for your job type. For example, not all job types offer bandwidth limiting, and not all jobs that do offer bandwidth limiting offer bandwidth scheduling. Use the **Set Options** page in the console as a guide to help you determine which job options are applicable to your job type.

Bandwidth

Set Options Field	Equivalent JobOptions Class
Do not limit bandwidth	JobOptions.BandwidthOptions.Mode
Use a fixed limit	JobOptions.BandwidthOptions.Mode
Use a fixed limit, Preset bandwidth	JobOptions.BandwidthOptions.Specification.Key JobOptions.BandwidthOptions.Specification.Type

Set Options Field	Equivalent JobOptions Class
	JobOptions.BandwidthOptions.Specification.Value
Use a fixed limit, Bandwidth	JobOptions.BandwidthOptions.Limit
Use scheduled limits	JobOptions.BandwidthOptions.Mode
Use scheduled limits, New (or Edit), Daytime entry	JobOptions.BandwidthOptions.Entries.EntryType
Use scheduled limits, New (or Edit), Overnight entry	JobOptions.BandwidthOptions.Entries.EntryType
Use scheduled limits, New (or Edit), Day	JobOptions.BandwidthOptions.Entries .DaysOfWeek
Use scheduled limits, New (or Edit), Start time	JobOptions.BandwidthOptions.Entries.StartTime
Use scheduled limits, New (or Edit), End time	JobOptions.BandwidthOptions.Entries.EndTime
Use scheduled limits, New (or Edit), Preset bandwidth	JobOptions.BandwidthOptions.Specification.Key JobOptions.BandwidthOptions.Specification.Type JobOptions.BandwidthOptions.Specification.Value
Use scheduled limits, New (or Edit), Bandwidth	JobOptions.BandwidthOptions.Entries.Limit

Compression

Set Options Field	Equivalent JobOptions Class
Enable compression	JobOptions.CoreConnectionOptions.ConnectionStartParameters.CompressionLevel.Algorithm JobOptions.CoreConnectionOptions.ConnectionStartParameters.CompressionLevel.Level

Failover Identity

Set Options Field	Equivalent JobOptions Class
Apply source network configuration to the target	JobOptions.CoreMonitorOptions.ShouldPerformLanFailover (for files and folders jobs) JobOptions.SystemStateOptions.IsWanFailover (for full server jobs)
Failover server name	JobOptions.CoreMonitorOptions.MonitorConfiguration.ItemsToFailover
Add these addresses to the selected target adapter after failover	JobOptions.CoreMonitorOptions.MonitorConfiguration.ItemsToFailover
Retain target network configuration	JobOptions.CoreMonitorOptions.ShouldPerformLanFailover (for files and folders and application jobs) JobOptions.SystemStateOptions.IsWanFailover (for full server jobs) JobOptions.VRAOptions.IsWanFailoverEnabled (for virtual guest jobs)
Failover server name	JobOptions.CoreMonitorOptions.MonitorConfiguration.ItemsToFailover
Update DNS server	JobOptions.DnsOptions.Enabled
Credentials for	JobOptions.CoreMonitorOptions.MonitorConfiguration. ActiveDirectoryCredentials
These DNS servers will be updated during failover	JobOptions.DnsOptions.Domains.DnsServers
Update these source DNS entries with the corresponding target IP address	JobOptions.DnsOptions.Domains.IpAddresMappings
Update TTL	JobOptions.DnsOptions.Domains.ShouldUpdateTtl JobOptions.DnsOptions.Domains.IpAddressMappings.ShouldUpdateTtl JobOptions.DnsOptions.Domains.TtlValue JobOptions.DnsOptions.Domains.IpAddressMappings.TtlValue

Set Options Field	Equivalent JobOptions Class
	Set both pairs of values to the same value. Both locations of ShouldUpdateTtl and both locations of TtlValue should be the same value. If they are different, the DNS update will fail.

Failover Monitor

Set Options Field	Equivalent JobOptions Class
Total time to failure	JobOptions.CoreMonitorOptions.TotalTimeAllowed JobOptions.CoreMonitorOptions.UseTotalTimeAllowed
Consecutive failures	JobOptions.CoreMonitorOptions.UseTotalTimeAllowed JobOptions.CoreMonitorOptions.MonitorConfiguration.Addresses.MaxPingAttempts
Monitor on this interval	JobOptions.CoreMonitorOptions.MonitorConfiguration.Addresses.PingInterval
Network monitoring	JobOptions.CoreMonitorOptions.MonitorConfiguration.Addresses.PingMethods
Monitor these addresses	JobOptions.CoreMonitorOptions.MonitorConfiguration.Addresses.Address
Monitoring method	JobOptions.CoreMonitorOptions.MonitorConfiguration.Addresses.PingMethods
Failover trigger	JobOptions.CoreMonitorOptions.MonitorConfiguration.Trigger
Service monitoring	JobOptions.MonitoringOptions.ServiceMonitoringEnabled
Services to monitor	JobOptions.ApplicationOptions.MonitoredServices
Attempt to restart this service after each failure	JobOptions.MonitoringOptions.ServiceMonitoringOptions.StartService
Custom script monitoring	JobOptions.CoreMonitorOptions.MonitorConfiguration.Addresses.PingMethods
Script file	JobOptions.CoreMonitorOptions.MonitorConfiguration.ScriptMonitorName

Failover Options

Set Options Field	Equivalent JobOptions Class
Wait for user to initiate failover	JobOptions.CoreMonitorOptions.MonitorConfiguration.ProcessingOptions.UserInterventionRequired
Shutdown source server	JobOptions.FullServerFailoverOptions.ShutdownSourceServer (for full server jobs) JobOptions.VRAOptions.WorkloadCustomizationOptions.ShouldShutdownSource (for virtual guest jobs)
Change target ports to match source during failover	JobOptions.SystemStateOptions.ApplyPorts
Failover shares	JobOptions.CoreMonitorOptions.MonitorConfiguration.ItemsToFailover.Shares
Failover host name	JobOptions.CoreMonitorOptions.MonitorConfiguration.ActiveDirectoryOptions
Failback host name	JobOptions.CoreMonitorOptions.MonitorConfiguration.ActiveDirectoryOptions
Active Directory Credentials	JobOptions.CoreMonitorOptions.MonitorConfiguration.ActiveDirectoryCredentials
Scripts	JobOptions.CoreMonitorOptions.MonitorConfiguration.Scripts

Failover Services

Set Options Field	Equivalent JobOptions Class
Services to stop on source and start on target during failover and start on source and stop on target during failback	JobOptions.SystemStateOptions.ServicesToStopOptions

General

Set Options Field	Equivalent JobOptions Class
Job name	JobOptions.Name
Reverse virtual machine display name	JobOptions.VRAOptions.ReverseOptions.VmName
Select the virtual recovery appliance on the source ESX server that will be used when reversing the job	JobOptions.VRAOptions.SourceApplianceInfo.Address JobOptions.VRAOptions.SourceApplianceInfo.GuestUri
Send data to the reverse appliance using this route	JobOptions.VRAOptions.ReverseRoute
Select the datastore on the reverse ESX server that will hold the reverse virtual machine	obOptions.VRAOptions.ReverseOptions.VmPath

Mirror, Verify & Orphaned Files

Set Options Field	Equivalent JobOptions Class
Choose a comparison method and whether to mirror the entire file or only the bytes that differ in each file	JobOptions.CoreConnectionOptions.ConnectionStartParameters.MirrorParameters.MirrorComparisonCriteria. JobOptions.CoreConnectionOptions.ConnectionStartParameters.MirrorParameters.MirrorOperationOptions
Enable scheduled verification	JobOptions.CoreConnectionOptions.ConnectionStartParameters.Schedule.Verify.IsEnabled
Verify on this interval	JobOptions.CoreConnectionOptions.ConnectionStartParameters.Schedule.Verify.Interval
Begin immediately	JobOptions.CoreConnectionOptions.ConnectionStartParameters.Schedule.Verify.StartTime
Begin at this time	JobOptions.CoreConnectionOptions.ConnectionStartParameters.Schedule.

Set Options Field	Equivalent JobOptions Class
	Verify.StartTime
Report only	JobOptions.CoreConnectionOptions.ConnectionStartParameters.Schedule.Verify.Criteria
Report and mirror files	JobOptions.CoreConnectionOptions.ConnectionStartParameters.Schedule.Verify.Criteria JobOptions.CoreConnectionOptions.ConnectionStartParameters.Schedule.Verify.Options
Calculate size of protected data upon connection	JobOptions.CoreConnectionOptions.ConnectionStartParameters.MirrorParameters.MirrorOperationOptions
Delete orphaned files	JobOptions.CoreConnectionOptions.ConnectionStartParameters.MirrorParameters.MirrorOperationOptions

Network Adapter Options

Set Options Field	Equivalent JobOptions Class
Map source network adapter to target network adapters	JobOptions.SystemStateOptions.NicMappings

Network Route

Set Options Field	Equivalent JobOptions Class
Send data to the target server using this route	JobOptions.CoreConnectionOptions.TargetAddress

Path Mapping

Set Options Field	Equivalent JobOptions Class
Mappings	JobOptions.CoreConnectionOptions.PathTransformations
Bock target paths upon connection	JobOptions.CoreConnectionOptions.ConnectionStartParameters. IsPathBlockingEnabled

Replica Virtual Machine Configuration

Set Options Field	Equivalent JobOptions Class
Display name	JobOptions.VRAOptions.ReplicaApplianceInfo.DisplayName
Sockets	JobOptions.VRAOptions.ReplicaVmInfo.CPUs
Cores per socket	JobOptions.VRAOptions.ReplicaVmInfo.CorsPerProcessor
Processors	JobOptions.VRAOptions.ReplicaVmInfo.CPUs
Memory	JobOptions.VRAOptions.ReplicaVmInfo.Memory
Network adapter type	JobOptions.VRAOptions.Volumes.DiskProvisioningType
Replica virtual switch	JobOptions.VRAOptions.VirtualSwitchMapping.TargetVirtualSwitch
Set VLAN on Replica	JobOptions.VRAOptions.ReplicaNetworkInterfaceInfo.VLAN_ID
Replica VLAN	JobOptions.VRAOptions.ReplicaNetworkInterfaceInfo.VLAN_ID
Power on replica after failover	JobOptions.VRAOptions.WorkloadCustomizationOptions. PowerupReplicaAfterFailover

Replica Virtual Machine Location

Set Options Field	Equivalent JobOptions Class
Select the datastore on the target ESX server that will hold the replica virtual machine	JobOptions.VRAOptions.Volumes.VirtualDiskPath
Select the volume and folder on the target server that will hold the replica virtual machine	JobOptions.VRAOptions.Volumes.VirtualDiskPath
Full path where the replica virtual machine will be stored	JobOptions.VRAOptions.Volumes.VirtualDiskPath
Local File Volume	None. This is an interface control only so the console can display the appropriate fields depending on the selected option.
SMB Share	None. This is an interface control only so the console can display the appropriate fields depending on the selected option.
File server name	JobOptions.VRAOptions.Volumes.VirtualDiskPath
Share name	JobOptions.VRAOptions.Volumes.VirtualDiskPath
Folder	JobOptions.VRAOptions.Volumes.VirtualDiskPath

Replica Virtual Machine Network Settings

Set Options Field	Equivalent JobOptions Class
Use advanced settings for replica virtual machine network configuration	JobOptions.VRAOptions.IsWanFailoverEnabled
Replica IP addresses	JobOptions.VRAOptions.ReplicaNetworkInterfaceInfo.IPAddresses
Replica Default Gateways	JobOptions.VRAOptions.ReplicaNetworkInterfaceInfo.Gateways
Replica DNS Server addresses	JobOptions.VRAOptions.ReplicaNetworkInterfaceInfo.DnsServers

Replica Virtual Machine Volumes

Set Options Field	Equivalent JobOptions Class
Replica Disk Size	JobOptions.VRAOptions.Volumes.DesiredSize JobOptions.VRAOptions.ReverseOptions.Volumes.DesiredSize
Replica Disk Format	JobOptions.VRAOptions.Volumes.DiskProvisioningType JobOptions.VRAOptions.ReverseOptions.Volumes.DiskProvisioningType
Storage Controller	JobOptions.VRAOptions.Volumes.DiskControllerType
Target Volume Target Datastore	JobOptions.VRAOptions.Volumes.VirtualDiskPath JobOptions.VRAOptions.ReverseOptions.Volumes.VirtualDiskPath
Virtual Disk	None. This is an interface control only so the console can display the appropriate fields depending on the selected option.
Pre-existing Disk Path	JobOptions.VRAOptions.Volumes.PreexistingDiskPath
Create disks match source	None. This is an interface control only so the console can display the appropriate fields depending on the selected option.
Create disks per volume	None. This is an interface control only so the console can display the appropriate fields depending on the selected option.
Disk Properties, Virtual disk	None. This is an interface control only so the console can display the appropriate fields depending on the selected option.
Disk Properties, Datastore	JobOptions.VRAOptions.DiskOptions.VirtualDiskPath
Disk Properties, Pre-existing disk path	JobOptions.VRAOptions.DiskOptions.PreexistingDiskPath
Disk Properties, Replica disk format	JobOptions.VRAOptions.DiskOptions.DiskProvisioningType
Disk Properties, Desired disk size	JobOptions.VRAOptions.DiskOptions.DesiredSizeInB
Volume Group Properties, Virtual disk	None. This is an interface control only so the console can display the appropriate

Set Options Field	Equivalent JobOptions Class
	fields depending on the selected option.
Volume Group Properties, Datastore	JobOptions.VRAOptions.LvmOptions.VolumeGroup.LogicalVolume.VirtualDiskPath JobOptions.VRAOptions.LvmOptions.VolumeGroup.PhysicalVolume.VirtualDiskPath
Volume Group Properties, Pre-existing disk path	JobOptions.VRAOptions.LvmOptions.VolumeGroup.PreexistingDisksPath
Volume Group Properties, Replica disk format	JobOptions.VRAOptions.LvmOptions.VolumeGroup.PhysicalVolume.DiskProvisioningType
Volume Group Properties, Physical volume maximum size	JobOptions.VRAOptions.LvmOptions.VolumeGroup.MaxPhysicalVolumeSize
Volume Group Properties, Volume Group Size	JobOptions.VRAOptions.LvmOptions.VolumeGroup.SourceVolumeGroupSize
Logical Volume Properties, Replica volume size	JobOptions.VRAOptions.LvmOptions.VolumeGroup.LogicalVolume.DesiredSize
Partition Properties, Virtual disk	None. This is an interface control only so the console can display the appropriate fields depending on the selected option.
Partition Properties, Datastore	JobOptions.VRAOptions.Volumes.VirtualDiskPath
Partition Properties, Pre-existing disk path	JobOptions.VRAOptions.Volumes.PreexistingDiskPath
Partition Properties, Replica disk format	JobOptions.VRAOptions.Volumes.DiskProvisioningType
Partition Properties, Replica volume size	JobOptions.VRAOptions.Volumes.DesiredSize

Reverse Protection and Routing

Set Options Field	Equivalent JobOptions Class
Send data to the target server using this route	JobOptions.CoreConnectionOptions.TargetAddress

Set Options Field	Equivalent JobOptions Class
Enable reverse protection	JobOptions.FullServerFailoverOptions.CreateBackupConnection
Select a reserved IP address on the source	JobOptions.SystemStateOptions.SourceReservedAddress
Select a reserved IP address on the target	JobOptions.SystemStateOptions.TargetReservedAddress

Scripts

Set Options Field	Equivalent JobOptions Class
Script file	JobOptions.CoreConnectionOptions.ConnectionStartParameters.ScriptPoints.Path JobOptions.CoreConnectionOptions.ConnectionStartParameters.ScriptPoints.Type
Arguments	JobOptions.CoreConnectionOptions.ConnectionStartParameters.ScriptPoints.Arguments
Allow script to interact with desktop	JobOptions.CoreConnectionOptions.ConnectionStartParameters.ScriptPoints.InteractionMode
Delay until script completes	JobOptions.CoreConnectionOptions.ConnectionStartParameters.ScriptPoints.ExecutionMode

Snapshots

Set Options Field	Equivalent JobOptions Class
Enable scheduled snapshots	JobOptions.CoreConnectionOptions.ConnectionStartParameters.SnapshotSchedule.IsEnabled
Take snapshots on this interval	JobOptions.CoreConnectionOptions.ConnectionStartParameters.SnapshotSchedule.Interval

Set Options Field	Equivalent JobOptions Class
Maximum number of snapshots	JobOptions.CoreConnectionOptions.ConnectionStartParameters.SnapshotSchedule.MaxNumberOfSnapshots
Begin immediately	JobOptions.CoreConnectionOptions.ConnectionStartParameters.SnapshotSchedule.StartTime
Begin at this time	JobOptions.CoreConnectionOptions.ConnectionStartParameters.SnapshotSchedule.StartTime

Staging Folder Options

Set Options Field	Equivalent JobOptions Class
Select additional folders from the source that need to be staged	JobOptions.FullServerFailoverOptions.AdditionalStagingFolders

Target Paths

Set Options Field	Equivalent JobOptions Class
Block target paths upon connection	JobOptions.CoreConnectionOptions.ConnectionStartParameters.IsPathBlockingEnabled

Target Route

Set Options Field	Equivalent JobOptions Class
Target route	JobOptions.CoreConnectionOptions.TargetAddress

Target Services

Set Options Field	Equivalent JobOptions Class
Services to leave running on the target during protection	JobOptions.TargetServicesOptions.FailoverServices

Test Failover

Set Options Field	Equivalent JobOptions Class
Use default replica virtual machine	JobOptions.VRAOptions.TestFailover.ReplicaDisplayName
Use alternate test replica virtual machine	JobOptions.VRAOptions.TestFailover.ReplicaDisplayName
Display name	JobOptions.VRAOptions.TestFailover.ReplicaDisplayName
Do not connect replica network adapter on test failover	JobOptions.VRAOptions.VirtualSwitchMappingTestFailover.TargetVirtualSwitch
Connect and map replica network adatpers on test failover	JobOptions.VRAOptions.VirtualSwitchMappingTestFailover.SourceVirtualSwitch
Replica Virtual Switch	JobOptions.VRAOptions.VirtualSwitchMappingTestFailover.TargetVirtualSwitch
Set VLAN on Replica	JobOptions.VRAOptions.ReplicaNetworkInterfaceInfo.VLAN_ID_TestFailover
Replica VLAN	JobOptions.VRAOptions.ReplicaNetworkInterfaceInfo.VLAN_ID_TestFailover
Replica Disk Format	JobOptions.VRAOptions.TestFailover.Volumes.DiskProvisioningType
Target Volume Target Datastore	JobOptions.VRAOptions.TestFailover.Volumes.VirtualDiskPath
Disk Properties, Datastore	JobOptions.VRAOptions.TestFailover.DiskOptions.VirtualDiskPath

Set Options Field	Equivalent JobOptions Class
Disk Properties, Replica disk format	JobOptions.VRAOptions.TestFailover.DiskOptions.DiskProvisioningType
Volume Group Properties, Datastore	JobOptions.VRAOptions.TestFailover.LvmOptions.VolumeGroup.LogicalVolume.VirtualDiskPath JobOptions.VRAOptions.TestFailover.LvmOptions.VolumeGroup.PhysicalVolume.VirtualDiskPath
Volume Group Properties, Replica disk format	JobOptions.VRAOptions.TestFailover.LvmOptions.VolumeGroup.PhysicalVolume.DiskProvisioningType
Partition Properties, Datastore	JobOptions.VRAOptions.TestFailover.Volumes.VirtualDiskPath
Partition Properties, Replica disk format	JobOptions.VRAOptions.TestFailover.Volumes.DiskProvisioningType
Delete test failover virtual disks	JobOptions.VRAOptions.TestFailover.DeleteVirtualDisks
Send data to the test failover server using this route	JobOptions.FullServerTestFailoverOptions.TestFailoverServerAddress
Test failover server	JobOptions.FullServerTestFailoverOptions.TestFailoverServerCredential.TestFailoverServerHostUri
Delete snapshots taken during test failover	JobOptions.FullServerTestFailoverOptions.DeleteSnapshots

Test Failover Scripts

Set Options Field	Equivalent JobOptions Class
Post-failover script, Script file	JobOptions.ApplicationOptions.TestPostFailoverScript
Post-failover script, Arguments	JobOptions.ApplicationOptions.TestPostFailoverScriptArguments
Pre-failback script, Script file	JobOptions.ApplicationOptions.TestPreFailbackScript
Pre-failback script, Arguments	JobOptions.ApplicationOptions.TestPreFailbackScriptArguments

Chapter 7 Server and job settings

The easiest way to view and change select server and job settings is through the Carbonite Replication Console. However, not all of the settings are available there, especially for Linux servers. To view and update the remaining settings, in addition to the settings available in the console, you will need to go to HKEY_LOCAL_MACHINE\SOFTWARE\NSI Software\Double-Take\CurrentVersion in the registry on a Windows server. For a Linux server, you can use DTSetup to modify the configuration settings or manually modify /etc/DT/DT.conf. For Windows or Linux, you can use the Carbonite PowerShell cmdlets Get-DtOption and Set-DtOptions.

The following table lists all of the Windows and Linux server and job settings, in decimal value.

- Windows server and job settings on page 493
- Linux server and job settings on page 527

Windows server and job settings

The following table lists all of the Windows server and job settings, in decimal value.

Carbonite products share the same set of server and job settings. You may only have a subset of the settings listed below depending on your Windows operating system and Carbonite product.

Carbonite Availability terminology is used in the following list. For example, failover is used for Carbonite Availability and cutover for Carbonite Migrate.

AcquireDataRetryLimit

Description—The length of time, in milliseconds, spent retrying a file read if there is a read error

Values—Any positive, integer value

Default—2000

Console setting—None

Service restart required—No

ActivationCode

Description—24-character Carbonite license key

Values—Unique value for each customer

Default—N/A

Console setting—Edit Server Properties page, Licensing section, Current license keys

Service restart required—No

AddOnCodes

Description—This setting is no longer used.

ArchiveLoopAttempts

Description—This setting is no longer used.

ArchiveLoopDelay

Description—This setting is no longer used.

AutoCalcEulaAccepted

Description—Used internally by Carbonite. Do not modify this entry.

AutoReconnect

Description—Specifies whether to reinstate the target connection(s) when the source machine is brought online after a source machine failure

Values—0 Do not reconnect, 1 Reconnect

Default—1

Console setting—Edit Server Properties page, Setup section, Automatically reconnect during source initialization

Service restart required—Yes

AutoRemirror

Description—Specifies whether to remirror when a source is brought online after an auto-disconnect

Values—0 Do not compare or send any files, 1 Compare file attributes and send the attributes and bytes that are different, 2 Do not compare files, just send all files (the entire file), 3 Compare file attributes and send the entire file for those that are different, 4 Compare file attributes and data and send the attributes and bytes that are different

Default—1

Console setting—Edit Server Properties page, Setup section, Behavior when automatically remirroring

Service restart required—No

AutoRemirrorRetry

Description—Specifies how often, in seconds, the source should check for connections that have been reconnected but still need to be remirrored

Values—any integer

Default—30

Console setting—None

Service restart required—No

AutoRetransmit

Description—Determines whether or not a source that has lost its connection with a target will attempt to reconnect to the target

Values—0 Do not attempt to reconnect, 1 Attempt to reconnect

Default—1

Console setting—None

Service restart required—No

BackupDir

Description—Location on the target of the backup of the protected data sets

Values—any valid path

Default—the location where the Carbonite files were installed

Console setting—None

Service restart required—No

CalculateByVolume

Description—Calculates the approximate size of a protected data set by using the size of the volume and subtracting the free space

Values—0 Disabled, 1 Enabled

Default—0

Console setting—None

Service restart required—Yes

CalculateOnConnect

Description—Specifies whether or not the amount of data to be mirrored should be calculated on connection

Values—0 Do not calculate on connection, 1 Calculate on connection

Default—1

Console setting—Edit Server Properties page, Source section, Calculate size of protected data upon connection

Service restart required—Yes

ChangeJournalState

Description—An internal setting for change journal tracking. Do not modify this setting.

ChangeJournalSystemState

Description—An internal setting for change journal tracking. Do not modify this setting.

ChecksumAll

Description—Indicates if a mirror, verify, or restore will ignore all attributes and perform a checksum calculation on all files

Values—0 Compare files by attribute, 1 Compare files by checksums

Default—1

Console setting—None

Service restart required—No

ClusterDir

Description—Location of a Microsoft Cluster Service installation, if it exists

Values—any valid path

Default—determined by the Microsoft Cluster Service installation

Console setting—None

Service restart required—No

ConnectionFile

Description—Name of the database file containing connection information

Values—any valid file name

Default—connect.sts

Console setting—None

Service restart required—No

CreateDumpOnAckErrors

Description—Enables additional logging for out of order acknowledgement errors

Values—0 Do not create a logging file, 1 Create a logging file

Default—0

Console setting—None

Service restart required—No

DataPath

Description—The location of the Carbonite file attribute, protected data set, connection, and schedule database files

Values—any valid path

Default—the location where the Carbonite files were installed

Console setting—None

Service restart required—No

DefaultAddress

Description—The default primary IP address in a multi-homed server

Values—any valid IP address that will act as your primary IP address for connecting the source to the target

Default—<null>

Console setting—Edit Server Properties page, General section, Default address

Service restart required—Yes

DefaultProtocol

Description—The default protocol

Values—2 IPv4 protocol only, 23 IPv4 and IPv6 protocols, 3 TDU (Throughput Diagnostics Utility)

Default—23

Console setting—None

Service restart required—Yes

DefaultReaderType

Description—Internal setting used for recoveries. Do not modify this setting.

DelayGCArbitration

Description—This setting is no longer used.

DelayGCConnection

Description—This setting is no longer used.

DiffMirrorHardLinkCleanup

Description—This setting is no longer used.

DisableAttributeReplication

Description—Specifies whether or not attributes (read-only, hidden, and so on) are replicated to the target

Values—0 Enable attribute replication, 1 Disable attribute replication

Default—0

Console setting—None

Service restart required—No

DriverJournalValid

Description—An internal setting for change journal tracking. Do not modify this setting.

DropOpOnAccessDeniedError

Description—Specifies whether or not operations are dropped or retried after an access denied error

Values—0 The operation will be retried, 1 The operation will be dropped

Default—1

Console setting—None

Service restart required—No

DropOpOnHandleError

Description—Determines if an additional attempt is made to access a file by a Microsoft API call if the Carbonite call fails.

Values—0 When opening a file using the Carbonite driver fails, attempt to open the file usingthe Microsoft Win32 API, 1 When opening a file using the Carbonite driver fails, skip the file and document it in the Carbonite log

Default—1

Console setting—None

Service restart required—No

Notes—If the value is set to 0 and the Win32 call also fails, Carbonite will skip the file and document it in the Carbonite log

DTSetupType

Description—Used by the Carbonite installation program to maintain the installation settings for an upgrade. Do not modify this setting.

DumpDiskQuotaIntervalMinutes

Description—Specifies how often, in minutes, a snapshot of the disk quota is taken as a backup in case the live registry is not usable at failover or cutover

Values—any integer

Default—240

Console setting—None

Service restart required—No

DumpHiveIntervalMinutes

Description—Specifies how often, in minutes, a snapshot of the registry is taken as a backup in case the live registry is not usable at failover or cutover

Values—any integer

Default—240

Console setting—None

Service restart required—No

EnableCRCCheck

Description—Indicates if Carbonite will perform a cyclic redundancy check between the source and target to identify corrupted packets

Values—0 Disabled, 1 Enabled

Default—0

Console setting—None

Service restart required—No

Notes—This option only needs to be set on the source server. However, if you will be restoring or reversing, where the roles of the servers are reversed, then you will need to set this option on the target as well.

EnableDHCP

Description—Indicates if Carbonite DHCP support is enabled

Values—0 Disabled, 1 Enabled

Default—1

Console setting—None

Service restart required—No

EnableEFSVerify

Description—Indicates if Carbonite will verify Microsoft encryption on the source before transmitting the encrypted file to the target

Values—0 Disabled, 1 Enabled

Default—0

Console setting—None

Service restart required—No

EnableFileOpenTracing

Description—Specifies if debug-level messages are enabled to trace all mirroring and replicated files that are opened

Values—0 Do not trace files that are opened, 1 Trace files that are opened

Default—0

Console setting—None

Service restart required—Yes

Notes—This option should only be enabled (1) for temporary, debug sessions as instructed by technical support.

EnableRootEncryption

Description—Specifies if the top-level folders of a protected data set are encrypted on the source, they will be encrypted on the target as well

Values—0 Disabled, 1 Enabled

Default—1

Console setting—None

Service restart required—No

Notes—If the top-level folders in a protected data set are not encrypted, disabling this option may obtain a small performance improvement.

EnableShortFileNameProcessing

Description—Indicates if Carbonite will correct any short file names created by the operating system on the target during a mirror. It will also correct any short file names created or renamed by the operating system on the target during replication.

Values—0 Do not correct any short file names on the target, 1 Correct short file names on the target

Default—0

Console setting—None

Service restart required—No

Notes—This setting only needs to be enabled on the target.

EnableSnapshots

Description—Specifies whether Carbonite snapshot functionality is enabled

Values—0 Carbonite snapshot functionality is disabled, 1 Carbonite snapshot functionality is enabled

Default—1

Console setting—None

Service restart required—Yes

Notes—This setting only impacts Carbonite snapshot functionality. If this setting is disabled, other snapshot software such as Microsoft Volume Shadow Copy will be not be impacted.

EnableTaskCmdProcessing

Description—Queues tasks inline with replication data

Values—0 Disable task command processing, 1 Enable task command processing

Default—0

Console setting—Edit Server Properties page, Setup section, Enable task command processing

Service restart required—No

EncryptNetworkData

Description—Encrypts Carbonite data before it is sent from the source to the target

Values—0 Disable data encryption, 1 Enable data encryption

Default—0

Console setting—Edit Server Properties page, General section, Encrypt network data

Service restart required—No

Notes—Both the source and target must be Carbonite encryption capable (Carbonite version 7.0.1 or later), however this option only needs to be enabled on the source or target in order to encrypt data. Keep in mind that all jobs from a source with this option enabled or to a target with this option enabled will have the same encryption setting. Changing this option will cause jobs to auto-reconnect and possibly remirror.

FailoverData1

Description—An internal setting for failover. Do not modify this setting.

FailoverData2

Description—An internal setting for failover. Do not modify this setting.

FileAccessRetry

Description—The number of times a failed driver call will be retried by the service.

Values—1 - 65535

Default—10

Console setting—None

Service restart required—No

FileQueueSize

Description—When a mirror is started, one thread reads from the disk and builds the file queue. Another set of threads reads files off of the queue and sends them to the target. This setting is the maximum size of the queue in entries. If you had 100 files to be mirrored and this

was set to 16 (the default value), the first thread would fill the queue to a maximum of 16 entries.

Values—1 - 65535

Default—16

Console setting—None

Service restart required—No

Notes—This value must be set prior to starting the mirror process. The higher the number, the more memory that is used.

ForceVerifyOnMirror

Description—Specifies if verification will be performed with every difference mirror

Values—0 Verification is not performed with every difference mirror, 1 Verification is performed with every difference mirror

Default—0

Console setting—None

Service restart required—No

HardlinkInterval

Description—This setting is no longer used.

HardLinkLogPath

Description—Specifies the location where hard links will be logged. If no path is specified, the location defined in LogDir will be used.

Values—any valid path

Default—None

Console setting—None

Service restart required—No

Note—This option is only used by servers running Carbonite version 8.0.x or earlier.

HBTTL

Description—Number of seconds without receiving a heartbeat before a remote machine is considered unavailable

Values—0 - 65535

Default—10

Console setting—None

Service restart required—No

HeartbeatIgnoreIPs

Description—This setting is no longer used.

HPQueueRatio

Description—Ratio of replication packets to one mirror packet

Values—1 - 65535

Default—5

Console setting—Edit Server Properties page, Source section, Number of replication packets per one mirror packet

Service restart required—No for future connections, Yes for the current connection

Notes—An HPQueueRatio of 5 allows Carbonite to dynamically change the ratio as needed based on the amount of replication data in queue. If you set a specific value other than the default (other than 5), the specified value will be used.

IgnoreAlternateStreamFiles

Description—Specifies alternate streams to skip during mirroring and replication

Values—a semi-colon separate list of stream names. The stream names are not case-sensitive

Default—none

Console setting—None

Service restart required—No

IgnoreArchiveBit

Description—Specifies if the archive bit is compared during verification

Values—0 Archive bit is compared during a verification, 1 Archive bit is not compared during a verification

Default—1

Console setting—None

Service restart required—No

IgnoreDeleteOps

Description—Specifies if file and directory delete operations will be replicated to the target

Values—0 Delete operations are replicated to the target, 1 Delete operations are not replicated to the target

Default—0

Console setting—None

Service restart required—No

IgnoreOpLockErrors

Description—Specifies how files that are locked open on the source are handled during mirroring

Values—0 Fail the mirror and record OpLock errors in the log. The job state will be set to mirror required, 1 Ignore the lock errors and continue the mirror. This option does not guarantee data integrity. There may be differences in the file that was locked.

Default—0

Console setting—None

Service restart required—No

IgnorePPPAddresses

Description—Identifies if Carbonite will use PPP (Point-to-Point Protocol) or SLIP (Serial Line Internet Protocol) adapters

Values—0 Carbonite will send out heartbeats across the PPP/SLIP adapter, 1 Carbonite will not send out heartbeats across the PPP/SLIP adapter

Default—1

Console setting—None

Service restart required—No

IgnoreSourceErrors

Description—This setting is no longer used.

IgnoreThumbnailStreams

Description—Specifies if thumbnails will be replicated to the target.

Values—0 Carbonite will mirror and replicate all data streams, 1 Carbonite will not mirror or replicate any data about the alternate data streams for thumbnail images. When comparing data for a verification or difference mirror,alternate data streams for thumbnails will not be reported as different.

Default—1

Console setting—None

Service restart required—If you change this value to 0, you must restart the Double-Take service in order for the Carbonite driver to begin sending all data stream information to the service. If you change this value to 1, you do not need to restart the service.

IgnoreWriteFailureOnTarget

Description—Specifies whether failures to write a file on the target are logged

Values—0 Log all write failures on the target, 1 or any larger integer indicates that number of write failures which will be ignored before starting to log the write failures

Default—0

Console setting—None

Service restart required—No

IncludeSysVolInfo

Description—Specifies whether the system volume information folder is mirrored and replicated

Values—0 Do not include the system volume information folder, 1 Include the system volume information folder

Default—0

Console setting—None

Service restart required—No

InstallPath

Description—Path specified during the Carbonite installation. Do not modify this entry.

InstallVersionInfo

Description—Installation number specified during the Carbonite installation. Do not modify this entry.

IntermediateQueueLimit

Description—Amount of memory, in KB, that may be allocated to the intermediate queue by the system memory manager when MemoryAllocatorMode is set to mixed mode (2).

Values—512-4194304

Default—65536

Console setting—None

Service restart required—Yes

KFAIOpenRetry

Description—Specifies the number of times an operation is retried if the driver return an error

Values—any valid integer

Default—10

Console setting—None

Service restart required—No

LanguageSelected

Description—Specifies the language of the verification log

Values—Depends on LanguagesSupported

Default—Language used during the installation

Console setting—Edit Server Properties page, Logging section, Language

Service restart required—Yes

LanguagesSupported

Description—Specifies the available languages for the verification log. Do not modify this setting.

LastModifiedReadDelay

Description—Specifies the length of time, in seconds, to wait before reading the last modified file time attribute

Values—any valid integer

Default—15

Console setting—None

Service restart required—No

Notes—This option is only used if SendLastModifiedTimeOnClose is disabled

LogAllOrphans

Description—Specifies whether success messages regarding orphan files are logged to the Carbonite log

Values—0 Do not log orphan file success messages to the Carbonite log, 1 Log orphan file success messages to the Carbonite log

Default—0

Console setting—None

Service restart required—No

LogDir

Description—The location of the Carbonite messages/alerts, verification, and statistics log files

Values—any valid path

Default—the location where the Carbonite files were installed

Console setting—Edit Server Properties page, Logging section, Logging folder

Service restart required—Yes

LogFile

Description—The name of the Carbonite messages/alerts log file

Values—any valid file name

Default—DTLog

Console setting—None

Service restart required—No

LogHardlinks

Description—This setting is no longer used.

LogMessageLevel

Description—Specifies the types of messages logged to the.dtl files

Values—0 No messages will be logged, 1 Only alert messages will be logged, 2 Alert and release messages will be logged, 3 Alert, release, and debug messages will be logged

Default—2

Console setting—None

Service restart required—No

MaxChecksumBlocks

Description—Specifies the number of checksum values retrieved from the target

Values—any integer

Default—32

Console setting—None

Service restart required—No

MaxConnections

Description—Number of network requests that can be processed simultaneously. Windows is limited to 5 simultaneous requests.

Values—0 - 65535

Default—5

Console setting—None

Service restart required—Yes

Notes—Carbonite recommends that you not change this value.

MaxLogFileSize

Description—Maximum size, in bytes, of any .dtl log file

Values—limited by available disk space

Default—5242880

Console setting—Edit Server Properties page, Logging section, Maximum size (under Messages & Alerts)

Service restart required—No

MaxLogPathname

Description—The maximum length of a file name (the entire volume\directory\filename including slashes, spaces, periods, extensions, and so on) that will be displayed in the Carbonite log file and the Windows Event Viewer. File names longer than the MaxDisplayablePath will be truncated and will be followed by an ellipsis (...).

Values—1-32760

Default—32760

Console setting—None

Service restart required—No

MaxNumberofLogFiles

Description—Maximum number of .dtl log files that can exist at one time. When Carbonite creates a new .dtl file, if this number is exceeded, the oldest .dtl file is deleted.

Values—1 - 999

Default—20

Console setting—Edit Server Properties page, Logging section, Maximum number of files

Service restart required—No

MaxOpBufferSize

Description—An internal setting for memory buffering. Do not modify this setting.

MaxRemoveOrphansOpSize

Description—Determines whether or not Carbonite will send over multiple orphan operations. Carbonite will send over the operations if a directory has more files than this number.

Values—0 - 131072

Default—1000

Console setting—None

Service restart required—No

MaxRetry

Description—A generic, application wide setting specifying the number of retry attempts for processes such as creating sockets, starting the service, and so on

Values—any integer

Default—5

Console setting—None

Service restart required—Yes

MaxWriteChunkSize

Description—Maximum merged op size (in bytes) used during replication

Values—1 - 131072

Default—65536

Console setting—None

Service restart required—No

MemoryAllocatorCallbackMode

Description—Determines what action is taken when the MemoryQueueToDiskThreshold is met

Values—0 Auto-disconnect processing is initiated when theMemoryQueueToDiskThreshold has been met. Connections will be reestablished when auto-reconnect occurs, 1 The Double-Take service stops pulling operations from the driver when theMemoryQueueToDiskThreshold has been met. The target will pause the source.The service will resume pulling operations when the target tells the source to resume, 2 The source and target begin queuing operations to disk.

Default—2

Console setting—None

Service restart required—Yes

MemoryQueueToDiskThreshold

Description—A percentage of QmemoryBufferMax that will trigger queuing to disk.

Values—any valid percentage

Default—75

Console setting—None

Service restart required—Yes

MinCompressionFileSize

Description—The minimum file size, in bytes, that will be compressed. Files smaller than this size will not be compressed.

Values—any file size

Default—1024

Console setting—None

Service restart required—No

MirrorChunkSize

Description—Block size, in bytes, used in the mirroring process

Values—1 - 1048576

Default—65536

Console setting—Edit Server Properties page, Source section, Size of mirror packets

Service restart required—No

Notes—A higher block size value gives you better throughput, but only to a certain point, then it starts using more memory (this has to do with the way memory is allocated and deallocated). A lower block size value produces slower throughput, but uses memory efficiently.

MirrorEncryptedFiles

Description—Specifies if Windows 200x encrypted files are mirrored

Values—0 Encrypted files are not mirrored, 1 Encrypted files are mirrored

Default—1

Console setting—None

Service restart required—No

MirrorOverwrite

Description—Determines if the mirror process overwrites existing files

Values—0 never overwrite, 1 always overwrite, 2 overwrite if older

Default—1

Console setting—None

Service restart required—No

MirrorQueueLimit

Description—Maximum number of mirror operations that can be queued on the source machine

Values—1 - 65535

Default—1000

Console setting—Edit Server Properties page, Source section, Maximum pending mirror operations

Service restart required—No

MirrorRootAttributes

Description—Specifies whether or not root permissions from the source are mirrored to the target

Values—0 Root permissions are not mirrored to the target, 1 Root permissions are mirrored to the target

Default—1

Console setting—None

Service restart required—No

MirrorZeroKFiles

Description—Specifies whether or not empty files, zero byte files, are included in a mirror

Values—0 Zero byte files are skipped and not mirrored to the target, 1 All files are mirrored to the target

Default—1

Console setting—None

Service restart required—No

Notes—If MirrorZeroKFiles is enabled (0), zero byte files are skipped during a full mirror, file differences mirror, and a verification with synchronization. Zero byte files that contain alternate data streams that are not empty, will still be skipped if MirrorZeroKFiles is enabled.

MoveOrphanedFiles

Description—This entry is no longer used.

MoveOrphansDir

Description—This entry is no longer used.

NetworkRetry

Description—Specifies the interval, in seconds, at which Carbonite will attempt to reconnect to the target

Values—any positive number

Default—10

Console setting—None

Service restart required—No

NetworkStatusInterval

Description—An internal setting for network communications. Do not modify this setting.

NetworkTimeout

Description—The maximum length of time, in seconds, to wait on a network connection. If data is not received over a network connection within the specified time limit, the connection is closed. During idle periods, Carbonite sends small amounts of keep-alive data at an interval 1/6 of the NetworkTimeout value to keep the socket from being inadvertently closed.

Values—any integer

Default—120

Console setting—None

Service restart required—No

NodeLockedLicenseKey

Description—An internal setting for licensing. Do not modify this setting.

NodeLockedServerInfo

Description—An internal setting for licensing. Do not modify this setting.

OpBufferMax

Description—Specifies the number of operations that can be stored in the memory queue prior to queuing to disk

Values—0 There is no limit to the number of operations that can be stored in the memory queue, 1 or any larger integer

Default—200000

Console setting—None

Service restart required—No

OpBuffersCount

Description—An internal setting for memory buffering. Do not modify this setting.

OpLogging

Description—Specifies whether operations from the Carbonite driver are logged

Values—0 Do not log operations, 1 Log operations

Default—0

Console setting—None

Service restart required—Yes

OutOfOrderDiff

Description—The maximum number of operations that can be out of order before the connection is paused

Values—any integer

Default—10

Console setting—None

Service restart required—No

Notes—The larger the value, the more memory the Double-Take service on the targe service will use.

Port

Description—Port connection for core Carbonite communications

Values—1025 - 65535

Default—6320

Console setting—Edit Server Properties page, General section, Port

Service restart required—Yes

ProductCode

Description—Used by the Carbonite installation program to maintain the installation settings for an upgrade. Do not modify this entry.

ProductName

Description—Used by the Carbonite installation program to maintain the installation settings for an upgrade. Do not modify this entry.

QJournalDir

Description—The location where the queue is stored.

Values—any valid path

Default—the location specified during the installation

Console setting—Edit Server Properties page, Queue section, Queue folder

Service restart required—No

Notes—For best results and reliability, you should select a dedicated, non-boot volume. The queue should be stored on a fixed, local NTFS volume. This location also stores the Carbonite driver pagefile.

QJournalFileSize

Description—The size, in MB, of each queuing transaction log file.

Values—any valid file size, up to 4095 MB

Default—5

Console setting—None

Service restart required—No

QJournalFreeSpaceMin

Description—The minimum amount of disk space, in MB, in the specified QJournalDir that must be available at all times.

Values—dependent on the amount of physical disk space available

Default—250

Console setting—Edit Server Properties page, Queue section, Minimum free disk space

Service restart required—No

Notes—The QJournalFreeSpaceMin should be less than the amount of physical disk space minus QJournalSpaceMax.

QJournalPreload

Description—The number of operations being pulled from the disk queue at one time. Do not modify this setting.

QJournalSpaceMax

Description—The maximum amount of disk space, in MB, in the specified QJournalDir that can be used for Carbonite queuing. When this limit is reached, Carbonite will automatically begin the auto-disconnect process.

Values—dependent on the amount of physical disk space available

Default—Unlimited

Console setting—Edit Server Properties page, Queue section, Limit disk space for queue

Service restart required—No

Notes—The unlimited setting allows the disk queue usage to automatically expand whenever the available disk space expands. Setting this option to zero (0) disables disk queuing. Even if you are using the unlimited option, Carbonite will only store 16,384 log files. If you are using the default 5MB file size, this is approximately 80GB of data. If you anticipate needing to be able to queue more data than this, you should increase the size of the log files.

QLogWriteThrough

Description—Specifies if the disk queues are write-through mode

Values—0 Disk queues are not write-through mode, 1 Disk queues are write-through mode

Default—0

Console setting—None

Service restart required—No

Notes—While write-through mode may decrease the frequency of auto-disconnects, it may also decrease the performance of the source server.

QMemoryBufferMax

Description—The amount of Windows system memory, in MB, that, when exceeded, will trigger queuing to disk.

Values—minimum 512, maximum is dependent on the server hardware and operating system

Default—1024

Console setting—Edit Server Properties page, Queue section, Amount of system memory to use

Service restart required—Yes

QueryOnQuorumFile

Description—Identifies if the Double-Take service will reopen closed files on the quorum drive

Values—0 The Double-Take service will not attempt to reopen a closed file on the quroum drive to get security descriptors or last modified times, 1 The Double-Take service will attempt to reopen a closed file on the quroum drive to get security descriptors or last modified times.

Default—1

Console setting—None

Service restart required—No

QueueSizeAlertThreshold

Description—The percentage of the queue that must be in use to trigger an alert message

Values—any valid percentage

Default—50

Console setting—Edit Server Properties page, Queue section, Alert at this queue usage

Service restart required—Yes

RemoveAllOrphans

Description—This entry is no longer used.

RemoveOrphansTime

Description—This entry is no longer used.

ReplicateNtSecurityByName

Description—Determines whether or not Carbonite replicates permissions and attributes assigned to local (non-domain) users and groups

Values—0 Do not replicate by name, 1 Replicate by name

Default—0

Console setting—Edit Server Properties page, Source section, Replicate NTFS security attributes by name

Service restart required—No

ReplicationDiskCheckScript

Description—Specifies the script to run if validation of the replication drive fails

Values—Any valid path and script file

Default—<null>

Console setting—None

Service restart required—No

ReplicationDiskCheckTimeOut

Description—Specifies the interval, in seconds, between validation checks when ReplicationDiskCheckSript is populated

Values—any integer

Default—300

GUI Setting—None

Service restart required—No

RepSetDBName

Description—Name of the database that contains protected data set information

Values—any valid file name

Default—DblTake.db

Console setting—None

Service restart required—No

RunDTInfoOnCutover

Description—Specifies if DTInfo is launched before a failover or cutover when protecting an entire server

Values—0 Do not launch DTInfo, 1 Launch DTInfo

Default—1

Console setting—None

Service restart required—No

RunScriptatSnaptime

Description—If a script is specified, the script is launched on the target before Carbonite executes any snapshots. The snapshot will not be executed until the script has completed. If the script returns an error, the snapshot will still execute.

Values—any valid path and script name

Default—<null>

Console setting—None

Service restart required—No

RunScriptInsteadofSnap

Description—Specifies if a script specified in RunScriptAtSnaptime is executed

Values—0 Execute script specified in RunScriptAtSnaptime, 1 Do not execute script specified in RunScriptAtSnaptime

Default—1

Console setting—None

Service restart required—No

SaveStatFile

Description—Determines if the statistic.sts (statistics logging) file is saved or ovewritten

Values—0 overwrite, 1 saved as statistic-old.sts

Default—1

Console setting—None

Service restart required—No

ScheduleFile

Description—Name of the database file that contains transmission scheduling information

Values—any valid file name

Default—Schedule.sts

Console setting—None

Service restart required—Yes

ScheduleInterval

Description—The number of seconds to wait before checking the transmission schedules to see if transmission should be started or stopped

Values—1 - 3600

Default—1

Console setting—None

Service restart required—Yes

SendDirLastModifiedTime

Description—Specifies if the last modified time for directories will be transmitted to the target during a difference mirror

Values—0 last modified time on directories will not be sent to the target, 1 last modified time on directories will be sent to the target

Default—1

Console setting—None

Service restart required—No

SendFileTimesOnCreate

Description—Specifies whether a file is accessed twice so that the file's creation time can be modified to match the source

Values—0 The Double-Take service will not access newly created files that have not been modified. These files on the target will have the date and time of when the file was created on the target, 1 The Double-Take service will access newly created files. These files on the target will have the same date and time as the source.

Default—0

Console setting—None

Service restart required—No

Notes—New files created on the source that have not been modified will have the date and time of when the file is created on the target. The date and time will be corrected to match the source's true file attributes when a remirror or verification modifies them to match the source or the file is modified by a user or application on the source. For example, if the source machine's clock is set to 2:00 PM and the target machine is set to 4:00 PM, a newly created file that has not been modified will have a time stamp of 4:00 PM when it is applied to the target. If this option is enabled (1), Carbonite will access the file twice, to correctly set the time to 2:00 PM to reflect the file's true attributes. If this option is disabled (0), Carbonite will not access the file twice, and the file will have the target time of 4:00 PM until it is modified (remirror, verification, or user or application update).

SendLastModifiedTimeOnClose

Description—Specifies that the last modified time attribute is sent when a file is closed

Values—0 Last modified time is sent when Carbonite has not received any additional operations for the file in the time period specified by LastModifiedReadDelay, 1 Last modified time is sent when a file is closed, which may not be immediately depending on system processing

Default—1

Console setting—None

Service restart required—No

Notes—If system processing delays (such as the system cache manager not flushing quickly enough) are causing delays in processing the last modified time, you may want to consider disabling this option (0).

ServerUUID

Description—Used internally by the Double-Take service to identify Carbonite connections and IP addresses used between servers

Values—Unique identifier generated by Carbonite

Default—Generated by Carbonite

Console setting—None

Service restart required—Yes

Notes—If you are certain that the server is not being used by any jobs, you can delete the ServerUUID. For example, you may want to delete the ServerUUID so that you can create an image of a server after installing Carbonite. A deleted ServerUUID will be re-created the next time the Double-Take service is started. Keep in mind, if you delete the ServerUUID and the server is being used by any jobs, you will have problems with all aspects of Carbonite including mirroring, replication, and failover.

ServicePriority

Description—The priority level at which the Double-Take service runs.

Values—2 normal priority, 3 high priority

Default—2

Console setting—None

Service restart required—Yes

Notes—The Double-Take service runs at normal priority by default. This option should not be modified, however, if the priority is raised to high (3), it can be done through Windows Task Manager.

ServicesToKeepRunning

Description—Services that will not be stopped on the target

Values—Semi-colon separated list of service names

Default—<null>

Console setting—Set Options page, Target Services section, Services to leave running on the target server during protection

Service restart required—No

Notes—You can specify the service name using the service executable file name or the service display name. There is no need to use quotation marks, even if the names have spaces in them. Only separate the names by a semi-colon (;).

ServiceStopState

Description—Used internally by the Double-Take service. Do not modify this entry.

ShortFileNameScanIntervalMinutes

Description—Specifies how often, in minutes, the registry is scanned for short file names

Values—any valid integer

Default—240

Console setting—None

Service restart required—No

ShutdownRebootTimeoutMinutes

Description—Specifies the amount of time, in minutes, to wait for the source to shutdown during failover or cutover

Values—any valid integer

Default—5

Console setting—None

Service restart required—No

ShutdownTimeout

Description—The amount of time, in seconds, for the service to wait prior to completing the shutdown so that Carbonite can persist data on the target in an attempt to avoid a remirror when the target comes back online

Values—any valid number of seconds where 0 (zero) indicates waiting indefinitely and any other number indicates the number of seconds

Default—0

Console setting—Edit Server Properties page, Setup section, Time allowed to complete shutdown operations

Service restart required—No

Notes—This setting only controls the service shutdown from the Carbonite clients. It does not control the service shutdown through a reboot or from the Service Control Manager.

SkipCompressionFileExt

Description—A period delimited list of file types that are not compressed, even if compression is enabled.

Values—any period delimited list of file types

Default—mp3.exe.wmv.wma.qt.mpg.mpeg.zip.jpg.jpeg.tiff.tar.rar.cab

Console setting—None

Service restart required—No

SnapshotType

Description—Specifies the type of snapshot that Carbonite takes

Values—0 Create a client-accessible or non-client-accessible snapshot based on the job type , 1 Always create a client-accessible snapshot, 2 Always create a non-client-accessible snapshot

Default—0

Console setting—None

Service restart required—No

SNMPTrace

Description—An internal setting for SNMP on Windows 2003 servers. Do not modify this setting.

SourcePendingAcks

Description—The number of operations received by the target queue in which the source is waiting for a response

Values—100 - 20,000

Default—2000

Console setting—None

Service restart required—No

SSMShutdownServices

Description—Used by full server jobs to determine services to shutdown during failover or cutover. Do not modify this entry.

StartupScript

Description—Used by full server jobs to control the post-failover script after reboot after failover. Do not modify this entry.

StatsDriverLogFlags

Description—Indicates which driver statistics are logged to the Carbonite log

Values—0 No driver statistics are logged, 1 State, 2 Operations, 4 Paging, 8 Timing

Default—0

Console setting—None

Service restart required—Yes

Notes—Use the sum of various values to log multiple driver statistics. For example, a setting of 5 would log paging and state statistics. A setting of 7 would log paging, operations, and state statistics. A setting of 15 would log all driver statistics.

StatsFileName

Description—Default file for logging statistics

Values—any valid file name

Default—statistic.sts

Console setting—Edit Server Properties page, Logging section, Filename (under Statistics)

Service restart required—No

StatsLoggingOn

Description—Specifies if Carbonite logs statistics at startup

Values—0 Stats logging does not start when Carbonite starts, 1 Stats logging starts when Carbonite starts

Default—0

Console setting—Edit Server Properties page, Setup section, Setup Options, Log statistics automatically

Service restart required—No

StatsMaxFileSize

Description—Maximum size, in MB, for the statistic.sts file

Values—limited by available disk space

Default—10485760

Console setting—Edit Server Properties page, Logging section, Maximum size (under Statistics)

Service restart required—No

StatsWriteInterval

Description—Interval, in minutes, in which statistics are written to the statistic.sts file

Values—0 - 65535

Default—5

Console setting—Edit Server Properties page, Logging section, Write interval

Service restart required—No

SystemMemoryLimit

Description—Set by the Double-Take service, each time it is started, to record the amount of available memory.

TargetPaused

Description—Internal setting that indicates if the target machine is paused. Do not modify this setting.

TargetPausedVirtual

Description—Internal setting that indicates which target machines are paused. Do not modify this setting.

TCPBufferSize

Description—Size of the TCP/IP buffer in bytes.

Values—4096-7500000

Default—375000

Console setting—None

Service restart required—Yes

Notes—The default setting creates a TCP window that will accommodate most environments. In most environments, this value will not need to be adjusted. However, if your Carbonite network has a long end-to-end route and the throughput is not where you would expect it to be, then adjusting this parameter may have beneficial results. This value is the bandwidth delay product, which is calculated using the bandwidth of the network (in bits/second) times the round trip time (in seconds) between the two ends. Use the following recommended settings to improve Carbonite throughput performance.

- 100Mbit LAN—The setting should be around 37500.
- 1Gbit LAN—The setting should be around 375000.
- WAN—The setting should be around 130000.

While the calculations are fairly straight forward, the values that have been suggested are not exact because they depend on round trip time. Some improvements could be gained by adjusting these values either higher or lower. The value suited for your environment can best be determined through trial and error testing.

TempDir

Description—Temporary directory used when replicating Windows 200x encrypted files.

Values—Any valid path

Default—\Program Files\Carbonite\Replication\Temp

Console setting—None

Service restart required—No

TGApplyMntPntSecurity

Description—Applies security settings to the volume of a mount point instead of applying them to the directory that the mount point is mounted to.

Values—0 Security will be applied to the directory, 1 Security will be applied to the volume

Default—0

Console setting—None

Service restart required—Yes

Notes—This setting needs to be applied to the target server.

TGBlockOnConnect

Description—Blocks the target path for all connections, regardless of the source, so that the data cannot be modified

Values—0 Target paths are not blocked, 1 Target paths are blocked

Default—0

Console setting—None

Service restart required—No

TGCloseDelay

Description—The length of time, in milliseconds, a file is held open on the target

Values—0 - 2000

Default—1000

Console setting—None

Service restart required—No

Notes—If disk caching on the target is disabled either manually or by default (for example, by default on disks that host Active Directory database files), the target system may be slow during a mirror. If so, descreasing this setting to 100, 10, and 0 will result in incremental improvements, with 0 returning the system performance to normal.

TGDaysToKeepMovedFiles

Description—Specifies the length of time, in days, to keep moved files if TGMoveFilesOnDelete is enabled

Values—any valid integer

Default—0

Console setting—Edit Server Properties page, Target section, Remove deleted files after this number of days

Service restart required—No

TGDisableAttributeReplication

Description—Specifies whether or not the attributes compression, ACL, and file mask are written to the target during mirroring and replication

Values—0 Enable attribute replication 1 Disable attribute replication

Default—0

Console setting—None

Service restart required—Yes

TGExecutionRetryLimit

Description—The number of times an unfinished operation will be retried on the target before it is discarded. If this value is set to zero (0), an operation will never be discarded and will be retried on the target until it is applied.

Values—0 - 65536

Default—0

Console setting—None

Service restart required—No

TGFileAlloc

Description—Indicates that Carbonite allocates an entire file on the first write of a mirror operation

Values—0 Disabled 1 Enabled

Default—1

Console setting—None

Service restart required—No

Notes—To help eliminate file fragmentation on the target server, Carbonite should allocate the entire file first. With extremely large files, the file allocation may take a long time. Therefore, you may want to disable the file allocation. If you disable file allocation, you will have more fragmentation on the target disk.

TGMirrorCapacityHigh

Description—Maximum percentage of system memory that can contain mirror data before the target signals the source to pause the sending of mirror operations.

Values—2-75

Default—20

Console setting—Edit Server Properties page, Target section, Pause mirroring at this level

Service restart required—No

TGMirrorCapacityLow

Description—Minimum percentage of system memory that can contain mirror data before the target signals the source to resume the sending of mirror operations.

Values—1-75

Default—15

Console setting—Edit Server Properties page, Target section, Resume mirroring at this level

Service restart required—No

Notes—The maximum value forTGMirrorCapacityLow is either 75 or TGMirrorCapacityHigh, which ever is lower.

TGMoveFilesOnDelete

Description—Specifies whether files deleted on the source are actually moved to a different location on the target rather than being deleted on the target

Values—0 Files deleted on the source will be deleted on the target, 1 Files deleted on the source will be moved to a different location on the target

Default—0

Console setting—Edit Server Properties page, Target section, Moved deleted files to this folder

Service restart required—No

Notes—If this option is enabled, the deleted files will be moved to the location specified in TGMoveFilesPath.

TGMoveFilesPath

Description—Specifies where deleted files on the source are being moved to on the target

Values—any valid path

Default—<null>

Console setting—Edit Server Properties page, Target section, Moved deleted files to this folder

Service restart required—No

TGMoveFilesSingleDirectory

Description—Specifies if deleted files that will be moved on the target (see **TGMoveFilesOnDelete**) will be moved to a single directory structure

Values—0 Use the same directory structure on the target as the source to store deleted files, 1 Use a single directory structure on the target to store deleted files

Default—0

Console setting—None

Service restart required—No

TGRetryLocked

Description—Minimum number of seconds to wait before retrying a failed operation on a target

Values—0-65536

Default—3

Console setting—Edit Server Properties page, Target section, Retry delay for incomplete operations

Service restart required—No

TGUnfinishedOpEvent

Description—Specifies whether or not unfinished operations on the target are logged to the Event Viewer

Values—0 Unfinished operation messages are not logged, 1 Unfinished operation messages are logged

Default—1

Console setting—None

Service restart required—No

TGWriteCache

Description—Specifies whether or not Carbonite uses the intermediate cache

Values—0 Bypass the intermediate cache and write directly to disk, 1 Do not bypass the intermediate cache

Default—1 for all other job types

Console setting—None

Service restart required—No

TGWriteFailureBeforeNotification

Description—Specifies the number of times an operation will be retried on the target before a notification is sent to update the target status

Values—0-1024

Default—10

Console setting—None

Service restart required—Yes

Notes—If you change the setting to 0, the notification will be disabled. Changing this option will only affect how the target status is displayed. To solve the underlying issue of why the operations are failing will require investigation into the Carbonite log files.

UpgradeCode

Description—Used by the Carbonite installation program to maintain the installation settings for an upgrade. Do not modify this entry.

UseChangeJournal

Description—Specifies if the Carbonite driver change journal is used to track file changes. If the source is rebooted, only the files identified in the change journal will be remirrored to the target. This setting helps improve mirror times.

Values—0 Do not track file changes and use the selected AutoRemirror option, 1 Track file changes and remirror only changed files on source reboot. If the change journal cannot be used, the selected AutoRemirror option will be used

Default—1

Console setting—Edit Server Properties page, Setup section, Mirror only changed files when source reboots

Service restart required—Yes

Notes—If you reboot your source into safe mode and changes are made to the protected data and then the source is rebooted normally, the Carbonite driver change journal will try but not be able to synchronize the source and target correctly because it was not loaded in safe mode. Therefore, you should manually start a difference mirror.

UseScheduledPause

Description—Used by Carbonite for internal schedule processing. Do not modify this setting.

VerifyLogAppend

Description—Specifies whether the DTVerify.log file will be appended to or overwritten

Values—0 Overwrite, 1 Append

Default—1

Console setting—Edit Server Properties page, Logging section, Append

Service restart required—No

VerifyLogLimit

Description—Maximum size of the DTVerify.log file in bytes

Values—limited by available hard drive space, up to 4 GB

Default—1048576

Console setting—Edit Server Properties page, Logging section, Maximum size (under Verification)

Service restart required—No

VerifyLogName

Description—Name of the verification log file

Values—any valid file name

Default—DTVerify.log

Console setting—Edit Server Properties page, Logging section, File name (under Verification)

Service restart required—No

VerifyRetryInterval

Description—The time, in minutes, between when one verification fails and a retry is scheduled to begin.

Values—any valid number

Default—3

Console setting—None

Service restart required—No

VerifyRetryLimit

Description—The number of time a verification will be retried.

Values—any valid number

Default—5

Console setting—None

Service restart required—No

VersionInfo

Description—The version of Carbonite that was installed. Do not modify this entry.

WatchDogFailureProcessDump

Description—Creates a troubleshooting dump file if the Carbonite driver stops running

Values—0 Do not create a dump file, 1 Create a dump file

Default—0

Console setting—None

Service restart required—No

WatchDogFailureScript

Description—Specifies the script to run if the Carbonite driver stops running

Values—Any valid path and script file

Default—<null>

Console setting—None

Service restart required—No

Linux server and job settings

The following table lists all of the Linux server and job settings, in decimal value.

Carbonite products share the same set of server and job settings. You may only have a subset of the settings listed below depending on your Linux operating system and Carbonite product.

Carbonite Availability terminology is used in the following list. For example, failover is used for Carbonite Availability and cutover for Carbonite Migrate.

ActivationCode

Description—24-character Carbonite license key

Values—Unique value for each customer

Default—N/A

Console setting—Edit Server Properties page, Licensing section, Current license keys

Service restart required—No

AdapterFlags

Description—Specifies the adapter to use when establishing a connection. This option should not be changed.

Values—2 Encryption, 4 Network Data Representation

Default—4

Console setting—None

Service restart required—No

Advertisement

Description—This setting is no longer used.

AllFailover

Description—Specifies which IP addresses to failover

Values—0 Failover only monitored IP addresses, 1 Failover all IP addresses

Default—1

Console setting—None

Service restart required—No

AllMustFail

Description—Specifies whether or not all IP addresses must fail for failover to take place

Values—0 Any IP address can fail, 1 All IP addresses must fail

Default—1

Console setting—None

Service restart required—No

AutoReconnect

Description—Specifies whether to reinstate the target connection(s) when the source machine is brought online after a source machine failure

Values—0 Do not reconnect, 1 Reconnect

Default—1

Console setting—None

Service restart required—Yes

AutoRemirror

Description—Specifies whether to remirror when a source is brought online after an auto-disconnect

Values—0 Do not compare or send any files, 1 Compare file attributes and send the attributes and bytes that are different, 2 Do not compare files, just send all files (the entire file), 3 Compare file attributes and send the entire file for those that are different, 4 Compare file attributes and data and send the attributes and bytes that are different

Default—1

Console setting—None

Service restart required—No

AutoRemirrorRetry

Description—Specifies how often, in seconds, the source should check for connections that have been reconnected but still need to be remirrored

Values—any integer

Default—30

Console setting—None

Service restart required—No

AutoRetransmit

Description—Determines whether or not a source that has lost its connection with a target will attempt to reconnect to the target

Values—0 Do not attempt to reconnect, 1 Attempt to reconnect

Default—1

Console setting—None

Service restart required—No

BackupDir

Description—Location on the target of the backup of the protected data sets

Values—any valid path

Default—the location where the Carbonite files were installed

Console setting—None

Service restart required—No

CalculateOnConnect

Description—Specifies whether or not the amount of data to be mirrored should be calculated on connection

Values—0 Do not calculate on connection, 1 Calculate on connection

Default—1

Console setting—None

Service restart required—Yes

CaseSensitiveRepSetQueries

Description—This entry is no longer used.

ChecksumAll

Description—Indicates if a mirror, verify, or restore will ignore all attributes and perform a checksum calculation on all files

Values—0 Compare files by attribute, 1 Compare files by checksums

Default—1

Console setting—None

Service restart required—No

Cleaner

Description—Specifies if a clean mirror will delete files on the target before mirroring

Values—0 Do not delete files before mirroring, 1 Delete files before mirroring

Default—0

Console setting—None

Service restart required—No

Notes—This option is only valid if you have this option enabled and use the clean option with the DTCL mirror command.

ClientLog

Description—This setting is no longer used.

ClientLogName

Description—This setting is no longer used.

ConnectionFile

Description—Name of the database file containing connection information

Values—any valid file name

Default—connect.sts

Console setting—None

Service restart required—No

DataPath

Description—The location of the Carbonite file attribute, protected data set, connection, and schedule database files

Values—any valid path

Default—the location where the Carbonite files were installed

Console setting—None

Service restart required—No

DefaultProtocol

Description—The default protocol

Values—1 IPv4 protocol only

Default—1

Console setting—None

Service restart required—Yes

DirUNetPort

Description—Port used for directed UDP communications

Values—1025 - 65535

Default—1505

Console setting—None

Service restart required—Source yes, Target no

DisableAttributeReplication

Description—Specifies whether or not attributes (user, group, or other permissions) are replicated to the target

Values—0 Enable attribute replication, 1 Disable attribute replication

Default—0

Console setting—None

Service restart required—Yes

EnablePerformanceTracking

Description—This entry will be used in the future.

EnableSparseFileMirroring

Description—Specifies if sparse files are mirrored

Values—0 Disable mirroring of sparse files, 1 Enable mirroring of sparse files

Default—1

Console setting—None

Service restart required—No

EnableTaskCmdProcessing

Description—Queues tasks inline with replication data

Values—0 Disable task command processing, 1 Enable task command processing

Default—0

Console setting—None

Service restart required—No

EnableVolumeLevelReplication

Description—Used by internally for full server jobs

EncryptionCipherFilter

Description—Encryption uses AES 256. Public key exchange uses industry-defined methods implemented by OpenSSL.

EncryptNetworkData

Description—Encrypts Carbonite data before it is sent from the source to the target

Values—0 Disable data encryption, 1 Enable data encryption

Default—0

Console setting—Edit Server Properties page, General section, Encrypt network data

Service restart required—No

Notes—Both the source and target must be Carbonite encryption capable (Carbonite version 8.0.0 or later), however this option only needs to be enabled on the source or target in order to encrypt data. Keep in mind that all jobs from a source with this option enabled or to a target with this option enabled will have the same encryption setting. Changing this option will cause jobs to auto-reconnect and possibly remirror.

ExtendedAttributes

Description—Specifies whether or not extended attributes are replicated to the target

Values—0 Extended attributes are not mirrored or replicated, 1 Extended attributes are mirrored and replicated

Default—0

Console setting—None

Service restart required—No

ExtensionNumber

Description—Used by the Carbonite log files.

FileQueueSize

Description—When a mirror is started, one thread reads from the disk and builds the file queue. Another set of threads reads files off of the queue and sends them to the target. This setting is the maximum size of the queue in entries. If you had 100 files to be mirrored and this

was set to 16 (the default value), the first thread would fill the queue to a maximum of 16 entries.

Values—1 - 65535

Default—16

Console setting—None

Service restart required—No

Notes—This value must be set prior to starting the mirror process. The higher the number, the more memory that is used.

HBExternalRate

Description—Number of seconds between heartbeats

Values—0 - 65535

Default—3

Console setting—None

Service restart required—No

Notes—Carbonite recommends a value that is less than 10 (see HBTTL). Zero (0) turns the heartbeats off.

HBInteralRate

Description—This entry is no longer used

HBLoopback

Description—This entry is no longer used.

HBTrace

Description—Specifies whether heartbeat debugging information is generated

Values—0 not generated, 1 Generated

Default—0

Console setting—None

Service restart required—No

HBTTL

Description—Number of seconds without receiving a heartbeat before a remote machine is considered unavailable

Values—0 - 65535

Default—10

Console setting—None

Service restart required—No

HPQueueRatio

Description—Ratio of replication packets to one mirror packet

Values—1 - 65535

Default—5

Console setting—None

Service restart required—No for future connections, Yes for the current connection

Notes—An HPQueueRatio of 5 indicates 5 replication packets to 1 mirror packet.

IgnoreDeleteOps

Description—Specifies if file and directory delete operations will be replicated to the target

Values—0 Delete operations are replicated to the target, 1 Delete operations are not replicated to the target

Default—0

Console setting—None

Service restart required—No

LoadSourceTarget

Description—Specifies the functionality of the loaded modules

Values—0 Neither the source nor target modules are loaded, 1 Only the source module is loaded, 2 Only the target module is loaded, 3 Both the source and target modules are loaded

Default—3

Console setting—None

Service restart required—Yes

LogAllOrphans

Description—Specifies whether success messages regarding orphan files are logged to the Carbonite log

Values—0 Do not log orphan file success messages to the Carbonite log, 1 Log orphan file success messages to the Carbonite log

Default—0

Console setting—None

Service restart required—No

LogDir

Description—The location of the Carbonite messages/alerts, verification, and statistics log files

Values—any valid path

Default—the location where the Carbonite files were installed

Console setting—None

Service restart required—Yes

LogFile

Description—The name of the Carbonite messages/alerts log file

Values—any valid file name

Default—DTLog

Console setting—None

Service restart required—No

LogMessageLevel

Description—Specifies the types of messages logged to the.dtl files

Values—0 No messages will be logged, 1 Only alert messages will be logged, 2 Alert and release messages will be logged, 3 Alert, release, and debug messages will be logged

Default—2

Console setting—None

Service restart required—No

MaxChecksumBlocks

Description—Specifies the number of checksum values retrieved from the target

Values—any integer

Default—32

Console setting—None

Service restart required—No

MaxConnections

Description—Number of network requests that can be processed simultaneously.

Values—0 - 65535

Default—5

Console setting—None

Service restart required—Yes

Notes—Carbonite recommends that you not change this value.

MaxLogFileSize

Description—Maximum size, in bytes, of any .dtl log file

Values—limited by available disk space

Default—5242880

Console setting—None

Service restart required—No

MaxNumberofLogFiles

Description—Maximum number of .dtl log files that can exist at one time. When Carbonite creates a new .dtl file, if this number is exceeded, the oldest .dtl file is deleted.

Values—1 - 999

Default—20

Console setting—None

Service restart required—No

MaxRemoveOrphansOpSize

Description—Determines whether or not Carbonite will send over multiple orphan operations. Carbonite will send over the operations if a directory has more files than this number.

Values—0 - 131072

Default—1000

Console setting—None

Service restart required—No

MaxRetry

Description—A generic, application wide setting specifying the number of retry attempts for processes such as creating sockets, starting the service, and so on

Values—any integer

Default—5

Console setting—None

Service restart required—Yes

MaxWriteChunkSize

Description—Maximum merged op size (in bytes) used during replication

Values—1 - 131072

Default—65536

Console setting—None

Service restart required—No

MemoryQueueToDiskThreshold

Description—A percentage of QmemoryBufferMax that will trigger queuing to disk.

Values—any valid percentage

Default—100

Console setting—None

Service restart required—Yes

MinCompressionFileSize

Description—The minimum file size, in bytes, that will be compressed. Files smaller than this size will not be compressed.

Values—any file size

Default—1024

Console setting—None

Service restart required—No

MirrorChunkSize

Description—Block size, in bytes, used in the mirroring process

Values—1 - 1048576

Default—65536

Console setting—None

Service restart required—No

Notes—A higher block size value gives you better throughput, but only to a certain point, then it starts using more memory (this has to do with the way memory is allocated and deallocated). A lower block size value produces slower throughput, but uses memory efficiently.

MirrorOverwrite

Description—Determines if the mirror process overwrites existing files

Values—0 never overwrite, 1 always overwrite, 2 overwrite if older

Default—1

Console setting—None

Service restart required—No

MirrorPrompting

Description—This entry is no longer used.

MirrorQueueLimit

Description—Maximum number of mirror operations that can be queued on the source machine

Values—1 - 65535

Default—1000

Console setting—None

Service restart required—No

MirrorZeroKFiles

Description—Specifies whether or not empty files, zero byte files, are included in a mirror

Values—0 Zero byte files are skipped and not mirrored to the target, 1 All files are mirrored to the target

Default—1

Console setting—None

Service restart required—No

Notes—If MirrorZeroKFiles is enabled (0), zero byte files are skipped during a full mirror, file differences mirror, and a verification with synchronization.

MissedPackets

Description—Specifies the number of requests sent by the target that go unanswered by the source before failover occurs, when using network responses to monitor for failover

Values—1 - 65535

Default—5

Console setting—None

Service restart required—No

MoveOrphanedFiles

Description—Specifies if orphaned files are deleted or moved to the directory specified by MoveOrphansDir

Values—1 Move, 0 Delete

Default—0

Console setting—None

Service restart required—No

MoveOrphansDir

Description—Indicates the name of the directory where orphaned files will be moved if MoveOrphanedFiles=1

Values—any valid path

Default—the location where the Carbonite files were installed

Console setting—None

Service restart required—No

NetPort

Description—Port connection for TCP communications

Values—1025 - 65535

Default—1500

Console setting—None

Service restart required—Yes

NetworkRetry

Description—Specifies the interval, in seconds, at which Carbonite will attempt to reconnect to the target

Values—any positive number

Default—10

Console setting—None

Service restart required—No

NetworkStatusInterval

Description—An internal setting for network communications. Do not modify this setting.

NetworkTimeout

Description—The maximum length of time, in seconds, to wait on a network connection. If data is not received over a network connection within the specified time limit, the connection is closed. During idle periods, Carbonite sends small amounts of keep-alive data at an interval 1/6 of the NetworkTimeout value to keep the socket from being inadvertently closed.

Values—any integer

Default—120

Console setting—None

Service restart required—No

Notes—If you are archiving files and it takes longer than the NetworkTimeout specified (for example, this may happen if the DTArchiveBin is located on an alternate volume), the archive operation will complete on the target, but the full file will not be changed to a link on the source because the source detected the network timeout.

NodeLockedLicenseKey

Description—An internal setting for licensing. Do not modify this setting.

OpBufferSize

Description—Specifies the number of operations that can be stored in the memory queue prior to queuing to disk.

Values—0 There is no limit to the number of operations that can be stored in the memory queue, 1 or any larger integer

Default—0

Console setting—None

Service restart required—No

PingFrequency

Description—Specifies, in seconds, how often a ping is sent to the source from a monitoring target

Values—1 - 65535

Default—5

Console setting—None

Service restart required—No

PreFailbackWait

Description—Specifies whether or not to wait for the target pre-failback script to complete before finishing a failback

Values—0 Do not wait, 1 Wait

Default—0

Console setting—None

Service restart required—No

PreFailoverWait

Description—Specifies whether or not to wait for the target pre-failover script to complete before finishing a failover

Values—0 Do not wait, 1 Wait

Default—0

Console setting—None

Service restart required—No

QJournalDir

Description—The location where the queue is stored.

Values—any valid path

Default—the location specified during the installation

Console setting—None

Service restart required—No

Notes—For best results and reliability, you should select a dedicated, non-boot volume. The queue should be stored on a dedicated, high-performance local volume (like SSD or high-speed HDD backed volume). The volume should not be used for any high I/O activity applications.

QJournalFileSize

Description—The size, in MB, of each queuing transaction log file.

Values—any valid file size, up to 4095 MB

Default—5

Console setting—None

Service restart required—No

QJournalFreeSpaceMin

Description—The minimum amount of disk space, in MB, in the specified QJournalDir that must be available at all times.

Values—dependent on the amount of physical disk space available

Default—250

Console setting—None

Service restart required—No

Notes—The QJournalFreeSpaceMin should be less than the amount of physical disk space minus QJournalSpaceMax.

QJournalPreload

Description—The number of operations being pulled from the disk queue at one time. Do not modify this setting.

QJournalSpaceMax

Description—The maximum amount of disk space, in MB, in the specified QJournalDir that can be used for Carbonite queuing. When this limit is reached, Carbonite will automatically begin the auto-disconnect process.

Values—dependent on the amount of physical disk space available

Default—Unlimited

Console setting—None

Service restart required—No

Notes—The unlimited setting allows the disk queue usage to automatically expand whenever the available disk space expands. Setting this option to zero (0) disables disk queuing. Even if you are using the unlimited option, Carbonite will only store 16,384 log files. If you are using the default 5MB file size, this is approximately 80GB of data. If you anticipate needing to be able to queue more data than this, you should increase the size of the log files.

QLogWriteThrough

Description—Specifies if the disk queues are write-through mode

Values—0 Disk queues are not write-through mode, 1 Disk queues are write-through mode

Default—0

Console setting—None

Service restart required—No

Notes—While write-through mode may decrease the frequency of auto-disconnects, it may also decrease the performance of the source server.

QMemoryBufferMax

Description—The amount of system memory, in MB, that, when exceeded, will trigger queuing to disk.

Values—minimum 32, maximum 4095

Default—256

Console setting—None

Service restart required—Yes

QueueSizeAlertThreshold

Description—The percentage of the queue that must be in use to trigger an alert message

Values—any valid percentage

Default—50

Console setting—None

Service restart required—Yes

RemapLink

Description—Specifies how Carbonite handles a soft link

Values—0 If a soft link exists in a replication set and points to a file or directory inside the replication set, the path contained in the link will retain its original mapping, 1 If a soft link exists in a replication set and points to a file or directory inside the replication set, Carbonite will remap the path contained in that link based on the Carbonite target path

Default—1

Console setting—None

Service restart required—No

RemoveAllOrphans

Description—Specifies if all orphan files will be removed or only those based on RemoveOrphanTime

Values—0 Remove orphans based on the entry RemoveOrphansTime, 1 Remove all orphans

Default—1

Console setting—None

Service restart required—No

RemoveOrphansTime

Description—Specifies the amount of time, in minutes, that must be expired before an orphan file is removed

Values—1 - 131072

Default—60

Console setting—None

Service restart required—No

ReplaceTarget

Description—Specifies whether or not to replace the target identity with the source identity during a failover

Values—0 Do not replace, 1 Replace

Default—0

Console setting—None

Service restart required—No

RepSetDBName

Description—Name of the database that contains protected data set information

Values—any valid file name

Default—DblTake.db

Console setting—None

Service restart required—No

RestoreOverwrite

Description—Determines if the restoration process overwrites existing files

Values—0 never overwrite, 1 always overwrite, 2 overwrite if older

Default—2

Console setting—None

Service restart required—No

RestorePrompting

Description—This entry is no longer used.

RestoreSpecialExecutableHandling

Description—Specifies if an alternate file is created and updated during a restoration for executables that are in use

Values—0 Do not use alternate files for executables that are in use, 1 Use alternate files for executables that are in use

Default—1

Console setting—None

Service restart required—No

SaveStatFile

Description—Determines if the statistic.sts (statistics logging) file is saved or ovewritten

Values—0 overwrite, 1 saved as statistic-old.sts

Default—1

Console setting—None

Service restart required—No

ScheduleFile

Description—Name of the database file that contains transmission scheduling information

Values—any valid file name

Default—Schedule.sts

Console setting—None

Service restart required—Yes

ScheduleInterval

Description—The number of seconds to wait before checking the transmission schedules to see if transmission should be started or stopped

Values—1 - 3600

Default—1

Console setting—None

Service restart required—Yes

ShareUpdateInterval

Description—Specifies how often, in minutes, the share file will be sent to the target

Values—1 - 65535

Default—60

Console setting—None

Service restart required—No

SkipCompressionFileExt

Description—A period delimited list of file types that are not compressed, even if compression is enabled.

Values—any period delimited list of file types

Default—
mp3.exe.wmv.wma.qt.mpg.mpeg.zip.jpg.jpeg.tiff.rar.cab.tgz.bz.bz2.z.pkg.sea.sit.sitx

Console setting—None

Service restart required—No

SmallFileThreshold

Description—Identifies the size of a small file. The entire file will be mirrored if the file size is below this threshold, thus improving mirror speeds.

Values—any integer

Default—65536

Console setting—None

Service restart required—No

SourcePendingAcks

Description—The number of operations received by the target queue in which the source is waiting for a response

Values—100 - 20,000

Default—2000

Console setting—None

Service restart required—No

StatsFileName

Description—Default file for logging statistics

Values—any valid file name

Default—statistic.sts

Console setting—None

Service restart required—No

StatsLoggingOn

Description—Specifies if Carbonite logs statistics at startup

Values—0 Stats logging does not start when Carbonite starts, 1 Stats logging starts when Carbonite starts

Default—0

Console setting—None

Service restart required—No

StatsMaxFileSize

Description—Maximum size, in MB, for the statistic.sts file

Values—limited by available disk space

Default—10485760

Console setting—None

Service restart required—No

StatsMaxObjects

Description—This entry is no longer used.

StatsPort

Description—Port used by DTStat to gather Carbonite statistics

Values—1025 - 65535

Default—1506

Console setting—None

Service restart required—No

StatsShmSize

Description—This entry is no longer used.

StatsWriteInterval

Description—Interval, in minutes, in which statistics are written to the statistic.sts file

Values—0 - 65535

Default—5

Console setting—None

Service restart required—No

SystemMemoryLimit

Description—Set by the Double-Take service, each time it is started, to record the amount of available memory.

TargetPaused

Description—Internal setting that indicates if the target machine is paused. Do not modify this setting.

TargetPausedVirtual

Description—Internal setting that indicates which target machines are paused. Do not modify this setting.

TCPBufferSize

Description—Size of the TCP/IP buffer in bytes.

Values—4096-7500000

Default—375000

Console setting—None

Service restart required—Yes

Notes—This is an operating system buffer, not a Carbonite buffer. If this option is set to zero (0), Linux kernel versions 2.6.7 or later can automatically tune this buffer setting for best server performance. Therefore, the recommended setting is 0 for automatic tuning, if you are using a version 2.6.7 or later Linux kernel. If you want to reduce or control network traffic, you can configure this option to a static size. The default is 375000 for a 1 GB network. Modifications should be relative to that speed using the calculation 37500 * network_speed_in_bits_per_ second / 100 Mbit.

TGCloseDelay

Description—The length of time, in milliseconds, a file is held open on the target

Values—0 - 2000

Default—1000

Console setting—None

Service restart required—No

Notes—If disk caching on the target is disabled either manually or by default (for example, by default on disks that host Active Directory database files), the target system may be slow during a mirror. If so, descreasing this setting to 100, 10, and 0 will result in incremental improvements, with 0 returning the system performance to normal.

TGExecutionRetryLimit

Description—The number of times an unfinished operation will be retried on the target before it is discarded. If this value is set to zero (0), an operation will never be discarded and will be retried on the target until it is applied.

Values—0 - 65536

Default—0

Console setting—None

Service restart required—No

TGMirrorCapacityHigh

Description—Maximum percentage of system memory that can contain mirror data before the target signals the source to pause the sending of mirror operations.

Values—2-75

Default—20

Console setting—None

Service restart required—No

TGMirrorCapacityLow

Description—Minimum percentage of system memory that can contain mirror data before the target signals the source to resume the sending of mirror operations.

Values—1-75

Default—15

Console setting—None

Service restart required—No

Notes—The maximum value forTGMirrorCapacityLow is either 75 or TGMirrorCapacityHigh, which ever is lower.

TGRetryLocked

Description—Minimum number of seconds to wait before retrying a failed operation on a target

Values—0-65536

Default—3

Console setting—None

Service restart required—No

TGThreadCount

Description—This setting is no longer used

TGUseExtendedQueue

Description—Specifies whether or not Carbonite uses the extended queue

Values—0 Use the extended queue, 1 Do not use the extended queue

Default—1

Console setting—None

Service restart required—No

TGWriteCache

Description—Specifies whether or not Carbonite uses the intermediate cache

Values—0 Bypass the intermediate cache and write directly to disk, 1 Do not bypass the intermediate cache

Default—0 for full server to ESX jobs, 1 for all other job types

Console setting—None

Service restart required—No

TGWriteFailureBeforeNotification

Description—Specifies the number of times an operation will be retried on the target before a notification is sent to update the target status

Values—0-1024

Default—10

Console setting—None

Service restart required—Yes

Notes—If you change the setting to 0, the notification will be disabled. Changing this option will only affect how the target status is displayed. To solve the underlying issue of why the operations are failing will require investigation into the Carbonite log files.

UNetPort

Description—Port connection for UDP communications

Values—1025 - 65535

Default—1500

Console setting—None

Service restart required—Yes

UpdateInterval

Description—Interval, in seconds, at which the Failover Control Center updates the monitored machines display

Values—1 - 9999

Default—1

Console setting—None

Service restart required—No

UserIntervention

Description—Specifies whether or not user intervention is required to initiate a failover

Values—0 User intervention is not required, 1 User intervention is required

Default—1

Console setting—None

Service restart required—No

UseShareFile

Description—Specifies whether to create and use a share file or to use the shares that are currently stored in the target memory

Values—0 Use the shares that are currently stored in the target memory, 1 Create and use a file containing the share information

Default—1

Console setting—None

Service restart required—No

VerifyLogAppend

Description—Specifies whether the DTVerify.log file will be appended to or overwritten

Values—0 Overwrite, 1 Append

Default—1

Console setting—None

Service restart required—No

VerifyLogLimit

Description—Maximum size of the DTVerify.log file in bytes

Values—limited by available hard drive space, up to 4 GB

Default—1048576

Console setting—None

Service restart required—No

VerifyLogName

Description—Name of the verification log file

Values—any valid file name

Default—DTVerify.log

Console setting—None

Service restart required—No

VerifyRetryInterval

Description—The time, in minutes, between when one verification fails and a retry is scheduled to begin.

Values—any valid number

Default—3

Console setting—None

Service restart required—No

VerifyRetryLimit

Description—The number of time a verification will be retried.

Values—any valid number

Default—5

Console setting—None

Service restart required—No

WarningPings

Description—This entry is no longer used.