New-DatabaseReleaseArtifact

New-DatabaseReleaseArtifact

Generates a Release Artifact containing deployment resources for a specific environment.

Syntax

```
New-DatabaseReleaseArtifact -Source <Object> -Target <Object> [-FilterPath <string>] [-IgnoreStaticData] [-
SQLCompareOptions <string>] [-SQLDataCompareOptions <string>] [-IgnoreAdditional] [-TransactionIsolationLevel
<TransactionIsolationLevel>] [-IncludeIdenticalsInReport] [-AbortOnWarningLevel <WarningSeverity>] [-
IgnoreParserErrors] [-SqlCmdVariables <hashtable>] [-CodeAnalysisSettingsPath <string>] [<CommonParameters>]
```

Description

The New-ReleaseArtifact cmdlet creates the database deployment resources containing all the information you need to update a target database. The output can be used by the Use-ReleaseArtifact cmdlet to update the target database, or by the Export-ReleaseArtifact cmdlet to export these resources for review.

Parameters

`-Source <System.Object>`

The schema you want to update databases to. This can be:

- a Database Connection object created by the New-DatabaseConnection cmdlet

- a database connection string

- a path for a NuGet package or .zip file. This must contain a scripts folder located at db\state

- a Database Build Artifact object produced by the New-DatabaseBuildArtifact cmdlet

- a path for a scripts folder, created by SQL Compare or from your SQL Source Control database repository

Aliases	None
Required?	true
Position?	named
Default Value	None
Accept Pipeline Input	false
Accept Wildcard Characters	false

`-Target <System.Object>`

A database schema you want to update from. This can be:

- a Database Connection object created by the New-DatabaseConnection cmdlet

- a database connection string

- a path for a NuGet package or .zip file. This must contain a scripts folder located at db\state

- a Database Build Artifact object produced by the New-DatabaseBuildArtifact cmdlet

- a path for a scripts folder, created by SQL Compare or from your SQL Source Control database repository

- a list containing one or more of the above

If you use a list, the cmdlet will check that everything in the list has the same database schema. If there are no differences in schema, the cmdlet will create the Database Release. If there are any differences in schema, the cmdlet will fail and inform you.

Aliases	None
Required?	true
Position?	named

Default Value	None
Accept Pipeline Input	false
Accept Wildcard Characters	false

-FilterPath <System.String>

The path to a .scpf filter file.

Overrides any Filter.scpf file present in the Source schema with an alternative filter file to be used when generating the Update.sql change script and in all schema comparisons.

This parameter will be ignored if the value specified is $null or empty.

Aliases	None
Required?	false
Position?	named
Default Value	None
Accept Pipeline Input	false
Accept Wildcard Characters	false

-IgnoreStaticData <System.Management.Automation.SwitchParameter>

If you use Red Gate SQL Source Control, you can flag tables that contain static data. If you specify a scripts folder or NuGet package as the value of Source and a database connection object or string as the value of Target, by default SQL Change Automation compares the data in these tables when generating the update script. Use this parameter to ignore static data when comparing the databases.

Aliases	None
Required?	false
Position?	named
Default Value	False
Accept Pipeline Input	false
Accept Wildcard Characters	false

-SQLCompareOptions <System.String>

Specifies the SQL Compare options to use when creating the update script and running pre- and post-deploy checks. SQL Change Automation applies a default set of options, listed below. To include additional options, specify a comma-delimited list of the option names (eg 'IgnoreComments, ObjectExistenceChecks'). To turn off a default option, precede the option name with a minus sign (eg '-ForceColumnOrder').

This parameter will be ignored if the value specified is $null or empty.

By default, the following Compare options are used:

- ConsiderNextFilegroupInPartitionSchemes

- DecryptPost2KEncryptedObjects

- DoNotOutputCommentHeader

- ForceColumnOrder

- IgnoreCertificatesAndCryptoKeys

- IgnoreDatabaseAndServerName

- IgnoreTSQLT

- IgnoreUsersPermissionsAndRoleMemberships

- IgnoreUserProperties

- IgnoreWhiteSpace

- IgnoreWithElementOrder

- IncludeDependencies

- ThrowOnFileParseFailed

- UseCompatibilityLevel

For more information about SQL Compare options, see http://www.red-gate.com/sca/ps/help/compareoptions.

Aliases	None
Required?	false
Position?	named
Default Value	None
Accept Pipeline Input	false
Accept Wildcard Characters	false

-SQLDataCompareOptions <System.String>

Specifies the SQL Data Compare options to use when creating the script for validation. To include additional options, specify a comma-delimited list of the option names (eg 'DisableAndReenableDDLTriggers, CompressTemporaryFiles').

Aliases	None
Required?	false
Position?	named
Default Value	None
Accept Pipeline Input	false
Accept Wildcard Characters	false

-IgnoreAdditional <System.Management.Automation.SwitchParameter>

Specifies whether to ignore additional objects in the target database. Enabling this prevents new and unexpected objects in the target being dropped - note that the checks and tests applied to this database may not have taken those objects into account and may therefore not be valid.

This parameter can't be used simultaneously with filter files.

Aliases	None
Required?	false
Position?	named
Default Value	False
Accept Pipeline Input	false
Accept Wildcard Characters	false

-TransactionIsolationLevel <RedGate.Versioning.Automation.Shared.Domain.TransactionIsolationLevel>

The isolation level for the transactions used in the update script. Permitted values are Serializable, Snapshot, RepeatableRead, ReadCommitted and ReadUncommitted. The default level is Serializable.

See http://msdn.microsoft.com/en-gb/library/ms173763.aspx for more details on transaction isolation levels.

Possible values: Serializable, Snapshot, RepeatableRead, ReadCommitted, ReadUncommitted

Aliases	None
Required?	false
Position?	named
Default Value	Serializable
Accept Pipeline Input	false
Accept Wildcard Characters	false

-IncludeIdenticalsInReport <System.Management.Automation.SwitchParameter>

By default, the change report will show the number of identical objects, but won't show the full SQL for each object. Use this parameter to include the full SQL of identical objects.

Aliases	None
Required?	false
Position?	named
Default Value	False
Accept Pipeline Input	false
Accept Wildcard Characters	false

-AbortOnWarningLevel <RedGate.Versioning.Automation.Shared.Domain.WarningSeverity>

Use this parameter to set the minimum warning level that will cause the release generation operation to abort.

Valid warning severity levels are:

- High

- Medium

- Low

- Information

- None (do not abort for any warnings)

The default setting is 'None'.

Possible values: Information, Low, Medium, High, None

Aliases	None
Required?	false
Position?	named
Default Value	None
Accept Pipeline Input	false
Accept Wildcard Characters	false

-IgnoreParserErrors <System.Management.Automation.SwitchParameter>

Tells the SQL Compare engine to ignore parser errors.

Aliases	None
Required?	false
Position?	named
Default Value	False
Accept Pipeline Input	false
Accept Wildcard Characters	false

-SqlCmdVariables <System.Collections.Hashtable>

Aliases	None
Required?	false
Position?	named
Default Value	None
Accept Pipeline Input	false
Accept Wildcard Characters	false

-CodeAnalysisSettingsPath <System.String>

Aliases	None
Required?	false
Position?	named
Default Value	None
Accept Pipeline Input	false
Accept Wildcard Characters	false

<CommonParameters>

This cmdlet supports the common parameters: -Verbose, -Debug, -ErrorAction, -ErrorVariable, -OutBuffer, and -OutVariable. For more information, see http://technet.microsoft.com/en-us/library/hh847884.aspx.

Inputs

The input type is the type of the objects that you can pipe to the cmdlet.

- **None.**
  You cannot pipe input to this cmdlet.

Return values

The output type is the type of the objects that the cmdlet emits.

- **RedGate.Versioning.Automation.Compare.Domain.ReleaseArtifacts.IReleaseArtifact**

Examples

---------- EXAMPLE 1 ----------

```
$staging = New-DatabaseConnection -ServerInstance "staging01\sql2017" -Database "Staging"
New-DatabaseReleaseArtifact -Source "C:\Work\buildArtifacts\DatabaseBuildArtifact.1.0.0.nupkg" -Target $staging
```

This example shows how to use a SQL Change Automation Build Artifact as a source.

The New-DatabaseReleaseArtifact cmdlet uses a SQL Change Automation Build Artifact, DatabaseBuildArtifact.1.0.0.nupkg, as the Source parameter. The package must have been generated from a SQL Change Automation project using the New-DatabaseBuildArtifact cmdlet

The Target parameter must be a connection string or a DatabaseConnection when using SQL Change Automation Projects.

---------- EXAMPLE 2 ----------

```
$staging = New-DatabaseConnection -ServerInstance "staging01\sql2014" -Database "Staging"
$buildArtifact = "C:\Work\buildArtifacts\MyDatabase.1.0.0.nupkg"

New-DatabaseReleaseArtifact -Source $buildArtifact -Target $staging
```

This example shows how to use a SQL Source Control Build Artifact to specify the source schema.

The New-DatabaseReleaseArtifact cmdlet uses a NuGet database package, MyDatabase.1.0.0.nupkg, as the Source parameter. The package must contain a scripts folder in the db\state sub-folder.

If the NuGet package includes tables that have been flagged as containing static data, this data will be included when generating the Release Artifact, unless the IgnoreStaticData parameter is specified.

You can also specify a NuGet package as the Target parameter. However, if you do this, static data won't be deployed.

---------- EXAMPLE 3 ----------

```
$staging = New-DatabaseConnection -ServerInstance "staging01\sql2014" -Database "Staging"
$test = New-DatabaseConnection -ServerInstance "test01\sql2014" -Database "Test" -Username "AutomationUser" -
Password "P@ssw0rd"

New-DatabaseReleaseArtifact -Source $test -Target $staging
```

This example show how to create the database deployment resources using two database connections.

The New-DatabaseConnection cmdlet creates two Database Connection objects, $staging and $test. $staging is used to connect to the Staging database on staging01\sql2014. $test is used to connect to the Test database on test01\sql2014.

The New-DatabaseReleaseArtifact cmdlet uses $test as the Source parameter, and $staging as the Target parameter. The cmdlet creates the database deployment resources that can be used to update the schema of Staging (and any database with the same schema as Staging) to match the schema of Test.

---------- EXAMPLE 4 ----------

```
$staging = New-DatabaseConnection -ServerInstance "staging01\sql2014" -Database "Staging"
$test = New-DatabaseConnection -ServerInstance "test01\sql2014" -Database "Test"

$options = "IgnoreComments, ObjectExistenceChecks, -ForceColumnOrder"

New-DatabaseReleaseArtifact -Source $test -Target $staging -SQLCompareOptions $options
```

This example shows how to specify the SQL Compare options that are used when generating the update script and running pre- and post-deploy checks.

In this example, the $options variable specifies that IgnoreComments and ObjectExistenceChecks should be included in addition to the default set of SQL Compare options. The minus sign before ForceColumnOrder indicates that this default option will be turned off.

---------- EXAMPLE 5 ----------

```
$staging = New-DatabaseConnection -ServerInstance "staging01\sql2014" -Database "Staging"
$test = New-DatabaseConnection -ServerInstance "test01\sql2014" -Database "Test"

$filter = "C:\Work\MyFilter.scpf"

New-DatabaseReleaseArtifact -Source $test -Target $staging -FilterPath $filter
```

This example shows how to use a specific filter file to include/exclude database objects when generating the update script, overriding any filter file present in the source schema.

MyFilter.scpf defines which database objects are considered when the update script is generated. If this update is used, the filters will also be used in pre- and post-schema comparisons.

---------- EXAMPLE 6 ----------

```
$staging = New-DatabaseConnection -ServerInstance "staging01\sql2017" -Database "Staging"
$sqlCmdVariables = @{MyVar = 'SomeVar'; MyOtherVar = '1234'}
New-DatabaseReleaseArtifact -Source "C:\Work\buildArtifacts\DatabaseBuildArtifact.1.0.0.nupkg" -Target $staging
-SqlCmdVariables $sqlCmdVariables
```

This example shows how to use a SQL Change Automation Build Artifact as a source with custom SQLCMD variables specified in the form of a hashtable.

Any necessary escaping will be performed, so take care not to double escape