Test-DatabaseConnection

Test-DatabaseConnection

Checks a DatabaseConnection object can connect to the database.

Syntax

```
Test-DatabaseConnection [-InputObject] <DatabaseConnection> [<CommonParameters>]
```

Description

The Test-DatabaseConnection cmdlet checks that the details in a DatabaseConnection object produced by the New-DatabaseConnection cmdlet can be used to connect to the specified database.

If Test-DatabaseConnection can't connect to the database, it raises an error.

After testing the connection, the cmdlet pipes out the input DatabaseConnection object. This allows you to use Test-DatabaseConnection in a chain of cmdlets.

Parameters

`-InputObject <RedGate.Versioning.Automation.Compare.SchemaSources.DatabaseConnection>`

The Database Connection object or database connection string that identifies the connection to test. See New-DatabaseConnection for details.

Aliases	None
Required?	true
Position?	0
Default Value	None
Accept Pipeline Input	true (ByValue)
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.

- **RedGate.Versioning.Automation.Compare.SchemaSources.DatabaseConnection**

  The Database Connection object or database connection string that identifies the connection to test. See New-DatabaseConnection for details.

Return values

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

- **RedGate.Versioning.Automation.Compare.SchemaSources.DatabaseConnection**

Examples

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

```
$connection = New-DatabaseConnection -ServerInstance "prod01\sql2014" -Database "Production" -Username
"AutomationUser" -Password "P@ssw0rd"
Test-DatabaseConnection $connection
```

This example shows how to use the Test-DatabaseConnection cmdlet to check that a Database Connection object can connect to a database.

The New-DatabaseConnection cmdlet creates a Database Connection object, $connection. This contains the details needed to connect to the database, Production, on prod01\sql2014. The Test-DatabaseConnection cmdlet then checks these details can be used to connect to Production.

If Test-DatabaseConnection can't connect to the database, you must check the connection details, and create a new Database Connection object with the correct details.

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

```
$connection = New-DatabaseConnection -ServerInstance "prod01\sql2014" -Database "Production" -Username
"AutomationUser" -Password "P@ssw0rd" | Test-DatabaseConnection
```

This example show how to pass a Database Connection object through the pipeline to the Test-DatabaseConnection cmdlet.

The New-DatabaseConnection cmdlet creates a Database Connection object. This is piped to the Test-DatabaseConnection cmdlet.

Test-DatabaseConnection checks that it can connect to the database, Production, using the Database Connection object. It then pipes the same Database Connection object out. This allows it to be assigned to the $connection variable.