Packaging databases

You can deploy SQL Server databases with Deployment Manager using database packages.

This article describes:

> **Applications**
>
> For information about creating application packages, see Packaging applications.

What database packages contain

Database packages contain database object creation scripts representing the state of a database (including any static data). Packages can also contain:

- upgrade scripts to upgrade a database from previous versions you have deployed with Deployment Manager
- creation scripts to create a database
- HTML documentation of the database

How Deployment Manager upgrades databases

Deployment Manager upgrades databases in one of two ways:

- **Dynamic upgrade (using SQL Compare during deployment)**
  By default, Deployment Manager upgrades the database automatically, using the SQL Compare engine.
- **Static upgrade (using an upgrade script)**
  If you created a database package using the SQL Automation Pack v1.0.4.2 or earlier, and chose the static deployment option, upgrade scripts are included in the database package.
  Deployment Manager checks if the package contains an upgrade script that's appropriate for the current deployment. If it does, Deployment Manager runs the script to upgrade the database. Otherwise, it performs a dynamic upgrade.

Deployment validation

Deployment Manager can validate a deployment by checking that the state of the database you are deploying to is identical to the state of the database in the package.

There are two types of validation:

- **Pre-deployment validation**
  Deployment Manager validates the deployment before it is performed.
  Pre-deployment validation is useful because it can detect database drift. Database drift occurs when changes are made to the database outside of a Deployment Manager deployment. If Deployment Manager performs a dynamic upgrade, these changes will be lost. If Deployment Manager performs a static upgrade, there may be errors when the upgrade script is executed.
  If database drift is detected, the validation fails. The database is not updated, and the next deployment steps are not performed.
- **Post-deployment validation**
  Deployment Manager validates the deployment after it has been performed.
  Post-deployment validation is useful because it can confirm that the deployment was successful. The validation will also detect any changes made to the database schema or static data outside of the Deployment Manager deployment (if a DDL or DML trigger ran during the deployment, for example).
  If the deployment was not successful, or any unexpected changes are detected, the validation fails. The database has been updated, but the next deployment steps are not performed.

Table-valued functions

> Table-valued functions are used in Deployment Manager 2.2.6 and later. They replace extended properties, which were used up to version 2.2.4.

When you deploy a database with Deployment Manager, or when you package a database using the SSMS add-in, the table-valued function **DeploymentManagerLastDeployment()** is added to the database. It has two columns, each with one row:

- **PackageName**
  The name of the package.

- **PackageVersion**
  The version number of the package.

When deploying a database with Deployment Manager, the properties are set with the package name and version of the package that you're deploying. When packaging databases using the SSMS add-in, the properties are set to the most recent package name and version you published.

The function is added to the *RedGateLocal* schema.

Extended properties

> Extended properties were used in Deployment Manager 2.2.4 and earlier. They have been replaced with table-valued functions in 2.2.6 and later.

When you first deploy a database with Deployment Manager, two extended properties are added to the database:

- **DeploymentManager Deployed Package ID**
  The name, or ID, of the database package.
- **DeploymentManager Deployed Package Version**
  The version number of the database package.

Deployment Manager uses these extended properties in future pre-deployment validations and static upgrades.

In future pre-deployment validations, the extended properties will be used to ensure that the database is validated against the correct package version.

In future static upgrades, the extended properties will be used to ensure that the correct upgrade script is used during the deployment.

The extended properties are updated automatically with each deployment.

> Before Deployment Manager 2.2.6, you cannot deploy SQL Azure databases with Deployment Manager because SQL Azure does not support extended properties.

Creating and publishing database packages

You can create and publish database packages from SQL Server Management Studio (SSMS) using the Deployment Manager add-in. For information, see:

- Publishing database packages from SSMS

If you're using TeamCity as your continuous integration server, see:

- Creating database packages in TeamCity

If you aren't using TeamCity, you can automate the creation of database packages with MSBuild:

- Creating database packages with MSBuild

Deploying database packages

To deploy database packages with Deployment Manager, you need to specify some variables in your project. For details, see:

- Deploying database packages