

SQL Server Technical Article

# Data-tier Applications in SQL Server 2008 R2

**Writer:** Alan Brewer, Roger Doherty, Prashant Choudhari, Shireesh Thota, Charlie Carson, Omri Bahat

**Technical Reviewer:** Jen Witsoe, S Muralidhar, Clifford Dibble, Sanjay Nagamangalam

**Published:**  February 2010

**Applies to:** SQL Server 2008 R2

**Summary:** Microsoft’s investments in application and multi-server management will help reduce the complexity around developing, deploying, and managing applications. A core concept of application and multi-server management in SQL Server is the data-tier application (DAC). A DAC is a self-contained unit of deployment that defines and bundles database objects, SQL Server instance objects that are associated with the database, and deployment requirements of an application.

Copyright

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2010 Microsoft Corporation. All rights reserved.

Microsoft, IntelliSense, PowerShell, SQL Server, Visual Studio, and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Table of Contents

[Data-tier Applications in SQL Server 2008 R2 1](#_Toc252546853)

[Introduction to data-tier applications 5](#_Toc252546854)

[What is a data-tier application? 5](#_Toc252546855)

[System requirements and general limitations 6](#_Toc252546856)

[Getting started 6](#_Toc252546857)

[Facilitating and automating the lifecycle of database projects 7](#_Toc252546858)

[Authoring data-tier applications 7](#_Toc252546859)

[Moving changes from development to production and viewing the DAC content 7](#_Toc252546860)

[Deploying a data-tier application for the first time 8](#_Toc252546861)

[Upgrading an existing data-tier application 9](#_Toc252546862)

[Extracting a DAC package from an existing database 10](#_Toc252546863)

[Registering a database to create a data-tier application 11](#_Toc252546864)

[Comparing a DAC package to a database 11](#_Toc252546865)

[Deleting a data-tier application 12](#_Toc252546866)

[Moving a data-tier application between instances of SQL Server 12](#_Toc252546867)

[Step by step - Data-tier applications in SQL Server Management Studio 13](#_Toc252546868)

[Extract Data-tier Application Wizard - Creating a DAC package from a database 13](#_Toc252546869)

[Installing a new DAC instance with the Deploy Data-tier Application Wizard 15](#_Toc252546870)

[Upgrading a data-tier application to a newer version using the wizard 18](#_Toc252546871)

[Deleting a data-tier application from an instance of SQL Server 22](#_Toc252546872)

[Register a database as a data-tier application 23](#_Toc252546873)

[Unpacking DAC packages to view the package content and deployment script 25](#_Toc252546874)

[Changing the application name or version of a deployed data-tier application 26](#_Toc252546875)

[Comparing DAC packages to databases 27](#_Toc252546876)

[Move a data-tier application between instances 28](#_Toc252546877)

[Using PowerShell to work with data-tier applications 30](#_Toc252546878)

[Extracting a DAC package from an existing database 30](#_Toc252546879)

[Registering and creating a data-tier application for an existing database 31](#_Toc252546880)

[Deploying a new data-tier application 32](#_Toc252546881)

[Upgrade a data-tier application to a newer version 33](#_Toc252546882)

[Deleting a data-tier application 35](#_Toc252546883)

[Developing and testing data-tier applications in Visual Studio 2010 37](#_Toc252546884)

[Introduction 37](#_Toc252546885)

[Creating and initializing a data-tier application 37](#_Toc252546886)

[Adding tables to the data-tier application project 41](#_Toc252546887)

[Adding views and stored procedures to the project and building the solution 44](#_Toc252546888)

[Creating a server selection policy to declare deployment intent 46](#_Toc252546889)

[Adding a post-deployment script to the project 48](#_Toc252546890)

[Configuring a data-tier application project for deployment 49](#_Toc252546891)

[Deploying a data-tier application 51](#_Toc252546892)

[Upgrading a data-tier application 52](#_Toc252546893)

[Limitations and support 53](#_Toc252546894)

[Objects supported in a data-tier application 53](#_Toc252546895)

[SQL Server version limitations 55](#_Toc252546896)

[DAC package size limitations 55](#_Toc252546897)

[Minimal permissions and security notes 55](#_Toc252546898)

[Disaster recovery and high-availability considerations 57](#_Toc252546899)

[Backup and restore 57](#_Toc252546900)

[Mirroring and log shipping 57](#_Toc252546901)

[Clustering 59](#_Toc252546902)

[Conclusion 59](#_Toc252546903)

[End notes 59](#_Toc252546904)

[Feedback 59](#_Toc252546905)

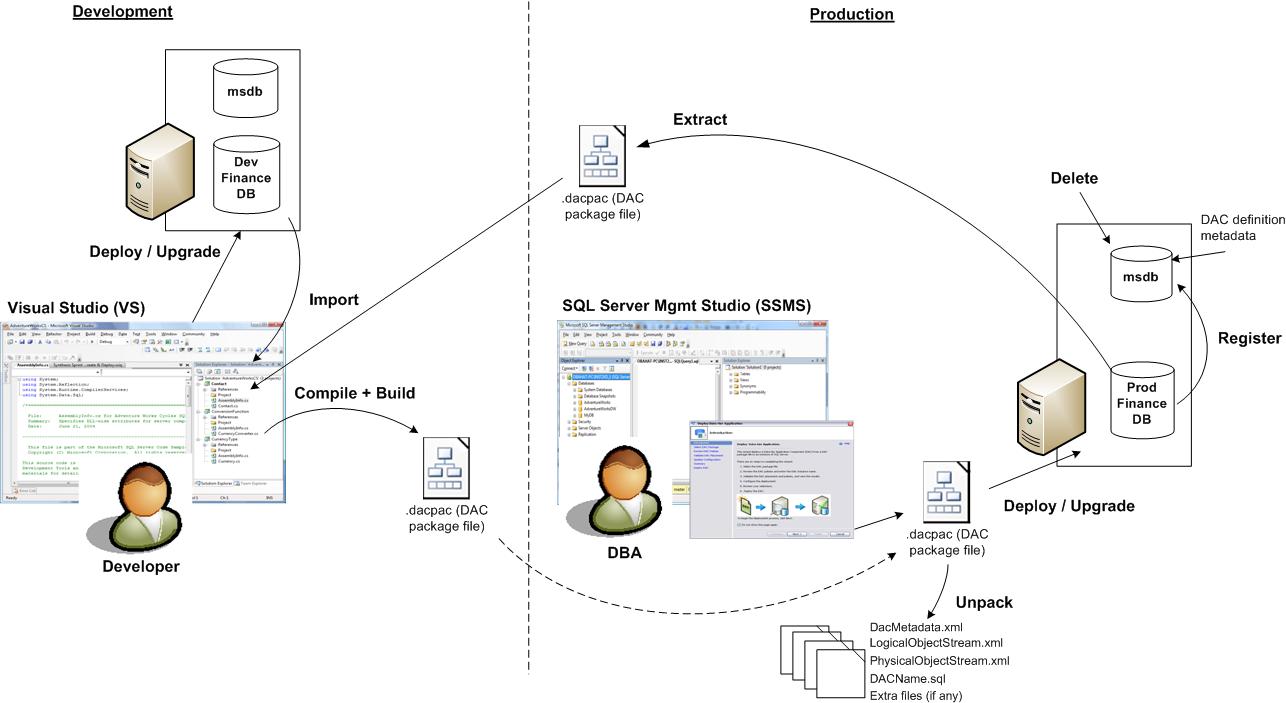
# Introduction to data-tier applications

Data-tier applications (DACs) are introduced in the Microsoft® SQL Server® 2008 R2 data management software and the Microsoft Visual Studio® 2010 development system to help users develop, deploy, and manage the data-tier portion of their applications more efficiently than before. Using DACs, developers and database administrators (DBAs) can automate and facilitate common operations and practices in the lifecycle of database systems.

## What is a data-tier application?

A DAC is a self-contained unit of deployment that enables data-tier developers and DBAs to package SQL Server objects, including database and instance objects, into a single entity called *DAC package.* Developers can *build* a DAC package using the Data-tier Application project system in Visual Studio, whereas SQL Server Management Studio (SSMS) users can *extract* a DAC and generate a DAC package file for an existing database.

A DAC package can be deployed to an instance of SQL Server to create a new *DAC instance*. The DAC deployment installs a new database on the instance, creates the database objects, and creates the logins associated with the users of the database. If a previous version of the DAC is already available, the DAC package can be used to *upgrade* the existing DAC instance to a newer version.



**Figure 1:** Data-tier application features in SQL Server 2008 R2 and Visual Studio 2010

In order to simplify the management of SQL Server environments, the health of DAC resources (for example, CPU, disk space) across multiple computers and instances can be viewed in a central interface called the *Utility Explorer* in SQL Server Management Studio. The Utility Explorer displays aggregated data and information, enabling DBAs to easily obtain utilization reports and statistics for their SQL Server installations. Furthermore, in order to customize CPU and disk space performance reports, DBAs can use *Utility Health Policies* to tune and control views showing the DAC resource consumption.

Note: The purpose of this document is to explain the development, deployment, and DAC features in the lifecycle of database projects, as shown in figure 1. For more information about the SQL Server Utility and the management of DACs in the Utility Explorer, see SQL Server Books Online.

## System requirements and general limitations

The initial release of data-tier application features in SQL Server 2008 R2 and Visual Studio 2010 is intended to serve departmental databases in the enterprise, as well as simple third-party (also referred to as ISV) applications. More explicitly, it is recommended that you use a data-tier application for a database that has the following characteristics:

* The database should contain up to a few gigabytes of data (for example, up to 10 GB). While this requirement is not strictly enforced by the data-tier application framework, working with larger databases may result in operational delays.
* The database meets the [limitation and support restrictions](#_Objects_supported_in) of data-tier applications.
* There are no specific hardware or software requirements for data-tier applications. However, to manage data-tier applications through the SQL Server Utility, users must consider the limitations imposed by the utility and managed instances, which are described in the appropriate topics in SQL Server Books Online.

## Getting started

Data-tier applications can be used to work with existing databases, or they can be used to implement new projects and releases. To get started, it is recommended that you create data-tier applications from existing systems by [registering](#_Registering_a_database) each production database as a data-tier application. Then, users can [extract](#_Extracting_a_DAC) each database to produce a DAC package and send the packages to the development team. From there on, developers use Visual Studio to [author data-tier changes](#_Authoring_data-tier_applications), package them appropriately, and forward the updated DAC packages to production. In turn, DBAs can [upgrade](#_Upgrading_an_existing) the production applications using automatic methods and tools that are provided by the data-tier application framework.

For new projects, developers can create a new data-tier application project in Visual Studio and [build](#_Adding_views_and) a DAC package for each application. Then, packages are forwarded to production DBAs for [deployment](#_Deploying_a_DAC). These and other lifecycle actions are illustrated in figure 1.

# Facilitating and automating the lifecycle of database projects

## Authoring data-tier applications

A data-tier application project in Visual Studio Professional comprises three key components: SQL Server objects, a *server selection policy,* and custom scripts (sometimes referred to as *additional* or *extra files*). In order to [get started](#_Creating_and_initializing) and define the schema of an application, developers can *import* a DAC package into the project system, or they can select an instance of SQL Server and import objects from an existing database. Then, users create and modify database objects and SQL Server instance objects that are associated with the database. For example, developers can use built-in designers to create tables and views. Programming users can utilize advanced code editing features (such as IntelliSense® technology, auto-completion, code snippets, and the Transact-SQL compiler) to write Transact-SQL code for stored procedures, functions, users, and logins.

After [composing the schema](#_Adding_tables_to), developers can create a DAC package and deploy the project to a target instance of SQL Server to test the release. In order to control deployments, a DAC package can contain a [server selection policy](#_Creating_a_server) that dictates prerequisite conditions on target instances. For example, a policy can restrict the deployment of a data-tier application to instances of SQL Server 2005 or later versions of SQL Server. The server selection policy helps developers to express and dictate deployment intents: Each time a DAC package is deployed, the policy is evaluated against the target to ensure that the instance meets the application requirements.

Custom deployment actions and data manipulation operations are also available in the data-tier application project through [custom scripts](#_Adding_a_post-deployment). Users can provide Transact-SQL scripts and include them in the DAC package. While the system does not automatically execute or reason over custom scripts, users can manually execute these scripts during [DAC deployment](#_Deploying_a_DAC) or [upgrade](#_Upgrading_an_existing). The custom scripts are extremely useful for populating and handling data or performing automatic actions before or after deployments; they are also helpful for dealing with objects that are not supported by a DAC (for more information, see [limitations of the DAC](#_Limitations_and_support)).

## Moving changes from development to production and viewing the DAC content

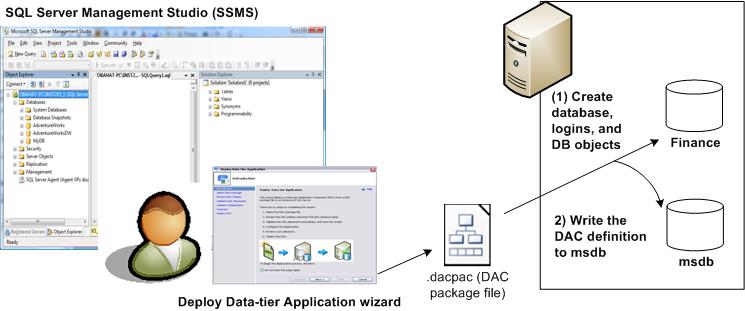
After a data-tier application is authored in Visual Studio and tested against SQL Server instances in the development environment, users can pack the content of the Visual Studio project and build a DAC package file (with the extension .dacpac). The package file can then be sent to DBAs in the test, staging, or production SQL Server environments for DAC deployment or upgrade, as illustrated in figure 1.

When receiving a DAC package from developers, DBAs running SQL Server Management Studio can double-click and *unpack* the file to view the package content and review the objects and code. In turn, three XML files and one SQL file are created on the file system, together with all pre/post-deployment and custom user-scripts (if any). The XML files hold the definition and metadata of the DAC. The SQL file is a Transact-SQL script that contains the DDL and CREATE statements for all DAC objects. This Transact-SQL script is executed during [DAC deployment](#_Deploying_a_DAC) or [upgrade](#_Upgrading_an_existing). The script is extremely useful for performing code reviews before changes are applied in production; it can also help [compare](#_Comparing_a_DAC) DAC objects to a target database before the DAC is upgraded.

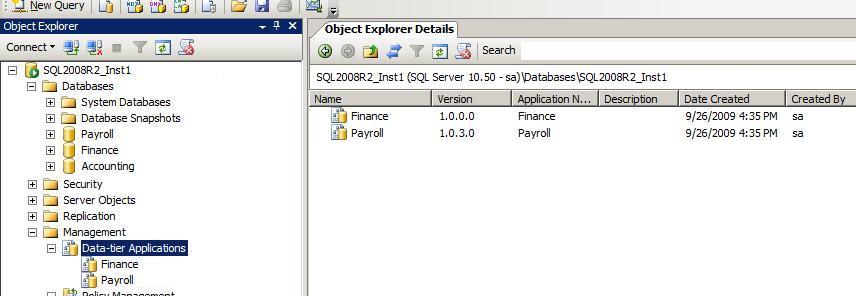
## Deploying a data-tier application for the first time

Say that a production DBA receives a DAC package from developers for a new application called *Finance*. In other words, assume that the Finance application and database were never installed in production before. To prepare for the deployment, the DBA can review the database objects and code by [unpacking](#_Unpacking_DAC_packages) the package file and opening the resulting Transact-SQL script named *Finance.sql* in SQL Server Management Studio. Then, the DBA selects a target instance of SQL Server that will host the database and objects.

Users can deploy a DAC package by using the Deploy Data-tier Application [Wizard](#_Installing_a_new) in SQL Server Management Studio, or through the Windows® [PowerShell](#_Deploying_a_new)® command-line interface. When deployment initiates, if the DAC contains a server selection policy, the policy is verified against the target instance [D01]. Then, the DAC content is installed: A new database, database objects, and all relevant server logins are created. After the database and objects are created, the DAC definition (that is, payload and metadata) is automatically registered and stored in the **msdb** system database. From that point on, the DAC and its properties can be viewed in SQL Server Management Studio under the Data-tier Applications node located under the Management node of the target instance in Object Explorer.



**Figure 2:** Deploying a data-tier application called Finance to an instance of SQL Server using SQL Server Management Studio



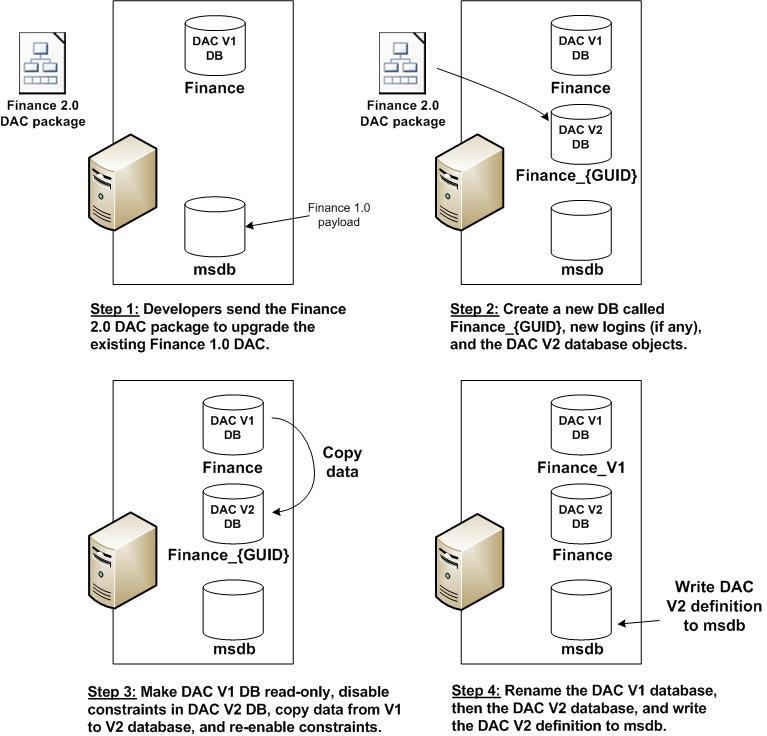
**Figure 3:** Viewing data-tier applications in Object Explorer and Object Explorer Details in SQL Server Management Studio

## Upgrading an existing data-tier application

Say that a Finance data-tier application with version V1 (or DAC V1 in short) has already been deployed in production. In addition, assume that developers made changes to DAC V1 and built a package file containing a newer version of the DAC (that is, DAC V2). After a production DBA receives the latter file, the DBA can upgrade the content of DAC V1 to V2. Before performing the upgrade, the DBA can [compare](#_Comparing_DAC_packages) the DAC package of the newer version to the database employed by DAC V1.

To upgrade a DAC, users can right-click the DAC V1 in Object Explorer in SQL Server Management Studio and then click **Upgrade Data-tier Application** to launch the [wizard](#_Upgrading_a_data-tier). Alternatively, users can upgrade DACs by invoking the appropriate [PowerShell](#_Upgrade_a_data-tier) commands. As in the case of DAC deployment, if the DAC V2 package contains a server selection policy, the policy conditions are first evaluated against the instance. Next, before the upgrade occurs, the DAC V1 database is compared to the definition of DAC V1 in **msdb** to see whether any changes were made to the database directly, without using the Data-tier Application Upgrade API (for example, by through the manual execution of SQL scripts to create database objects). If any changes are detected, the user must acknowledge that those changes may be lost during the DAC upgrade [D02].

Finally, the upgrade process begins: A new database is created with the DAC V2 schema and objects, as well as any logins that do not yet exist on the instance. Then, data is copied from the DAC V1 database to the database used by DAC V2. To complete the upgrade process, the DAC V1 database is renamed, the DAC V2 database takes the name of the original DAC V1 database, and the DAC definition is updated in **msdb** to reflect the changes and register DAC V2. To illustrate this process, the image in figure 4 shows the steps for upgrading the Finance DAC.



**Figure 4:** Step by step process of upgrading the Finance data-tier application from version V1 to V2

## Extracting a DAC package from an existing database

The Extract Data-tier Application [Wizard](#_Extract_Data-tier_Application) in SQL Server Management Studio and the corresponding [PowerShell](#_Extracting_a_DAC_1) methods enable developers and DBAs to select an existing database and generate a DAC package file (see figure 1) that holds a definition of the database, database objects, and all relevant server logins.

The extract DAC functionality is designed to achieve several goals and provide solutions for numerous scenarios in the application lifecycle: In order to initialize a data-tier application project in Visual Studio with the latest production schema, a DBA can extract a package file from the production database and send the file to developers. In turn, developers [import](#_Creating_and_initializing) the DAC package to start working with the Visual Studio project. In other applications, extracting a DAC according to a preconfigured schedule (for example, using an SQL job) enables DBAs to record schema snapshots for their SQL Server databases. Furthermore, DAC extraction is a key step in [comparing](#_Comparing_a_DAC_1) a DAC package to a database before upgrading a data-tier application.

## Registering a database to create a data-tier application

If a database is not part of a data-tier application, users can create a DAC *in-place*. This is done by registering the database as a data-tier application, by using the [Register Data-tier Application Wizard](#_Register_a_database) in SQL Server Management Studio or through [PowerShell](#_Registering_and_creating). As in the case of [extraction](#_Extract_Data-tier_Application), the registration process reads the content of the database, the objects, and logins, and it generates the DAC definition (in memory). Then, the DAC definition is written to the **msdb** system database and the new data-tier application can be viewed in the Object Explorer and Object Explorer Details panes in SQL Server Management Studio, as shown in figure 3.

Key benefits of registering a database as a data-tier application include the following:

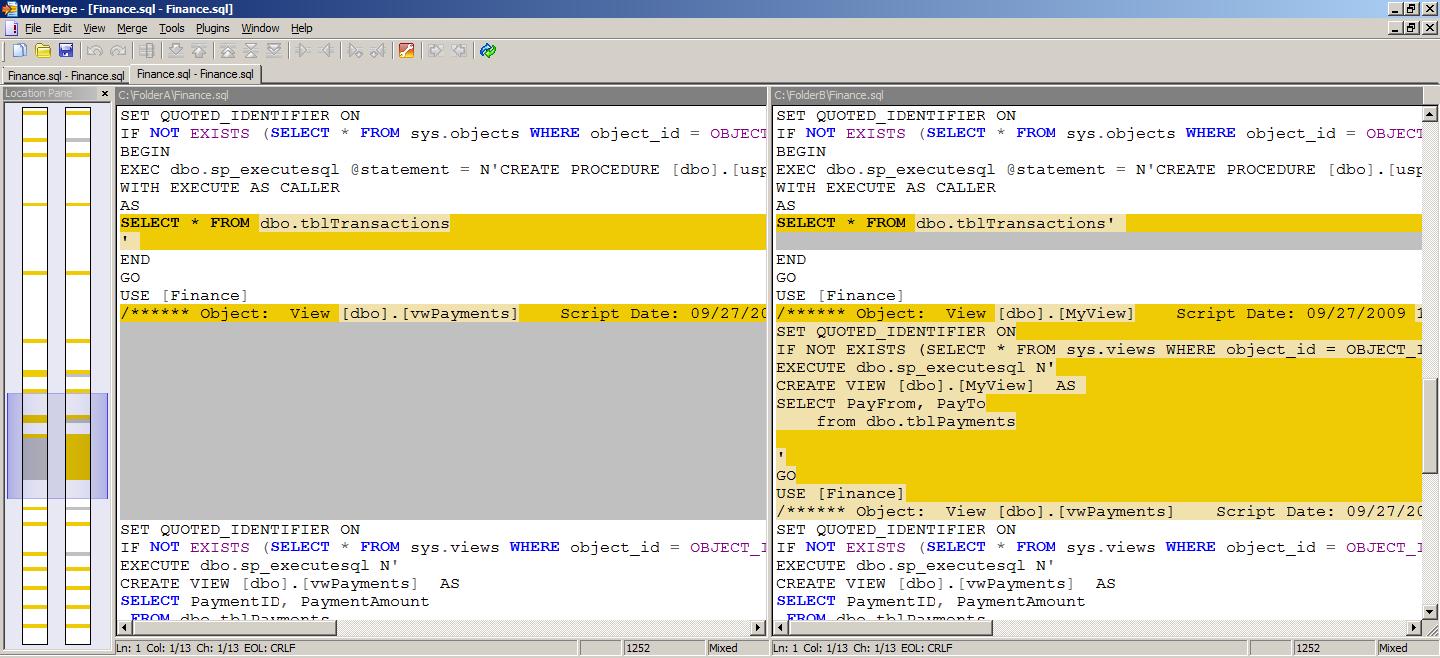
* If a database resides on a utility-managed instance (that is, an instance of SQL Server that is enrolled in the SQL Server Utility), the registration turns on the utility management features for the data-tier application and database.
* After a data-tier application is created, future versions of the application can be automatically upgraded using the [DAC upgrade](#_Upgrading_a_data-tier) method and API.

## Comparing a DAC package to a database

SQL Server Management Studio in SQL Server 2008 R2 does not provide built-in interfaces to compare DAC packages and databases. However, users can compare DAC packages to databases using SQL Server Management Studio in combination with commonly available software tools.

Say that version V1 of a DAC is running in production and that a DAC package with version V2 is available for [upgrade](#_Upgrading_a_data-tier). In order to compare the differences between V1 and V2 and view changes to the database and logins, users can perform the following actions:

1. Install any Windows file comparison utility, such as WinDiff or WinMerge.
2. Double-click the DAC V2 package file on a computer running SQL Server Management Studio to unpack the package content and files to a folder (say, a folder called V2Folder). Locate the file called *DACName*.sql in the folder, where *DACName* is the name of the data-tier application.
3. [Extract](#_Extract_Data-tier_Application) a data-tier application from the database used by DAC V1, to create a DAC package.
4. Unpack the DAC V1 package file created in step 3 and write the package file to some folder (for example, V1Folder). Locate the file called *DACName*.sql in this folder.
5. Use the file comparison utility from step 1 to compare the .sql file from step 2 and the file from step 4, as shown in figure 5.



**Figure 5:** Comparing the content of a DAC package file to a database using a file comparison utility

## Deleting a data-tier application

Data-tier applications can be deleted from an instance of SQL Server by removing the DAC definition and history records from the **msdb** system database, while leaving the DAC database on the instance untouched. As a result, the data-tier application will no longer appear in Object Explorer in SQL Server Management Studio. In addition, users can choose to detach or drop the DAC database. Either way, logins are not removed from the instance as part of the deletion process.

## Moving a data-tier application between instances of SQL Server

A data-tier application that resides on an instance of SQL Server consists of a database, logins that are associated with the database users, and the DAC definition and metadata in **msdb**. Consequently, data-tier applications can be manually moved from one instance of SQL Server to another (for example, from instance A to instance B) through some simple manual steps:

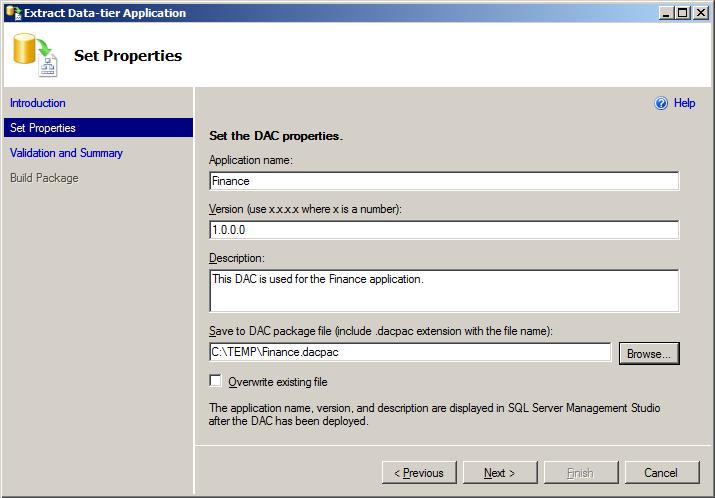
1. Create a full backup of the database on instance A, restore the database from backup on instance B while keeping the same database name. Then, [register](#_Register_a_database) the database as a data-tier application on instance B and use the Delete Data-tier Application [Wizard](#_Deleting_a_data-tier_1) to delete the DAC registration and drop the database from instance A. It should be understood that the data-tier application history in **msdb** in instance A will be lost during the move. Furthermore, if some logins exist on instance A but not on instance B, then those logins have to be created manually. Similar arguments apply to synchronizing the SID values for users and logins in instance B after the database is restored, which is a standard step in the backup-restore process.
2. Detach the database from instance A and attach it to instance B, and then [register](#_Register_a_database) the database as a data-tier application on instance B. Previous comments regarding **msdb** history records, as well as login creation or synchronization apply here as well.

# Step by step - Data-tier applications in SQL Server Management Studio

## Extract Data-tier Application Wizard - Creating a DAC package from a database

Users can operate the Extract Data-tier Application Wizard in order to generate and save a DAC package for a database. In this process, the wizard retrieves the content of the database and the logins associated with the database, and then it validates that a data-tier application can indeed be created. Then, the wizard writes the DAC package to the file system. To extract a DAC package, perform the following steps:

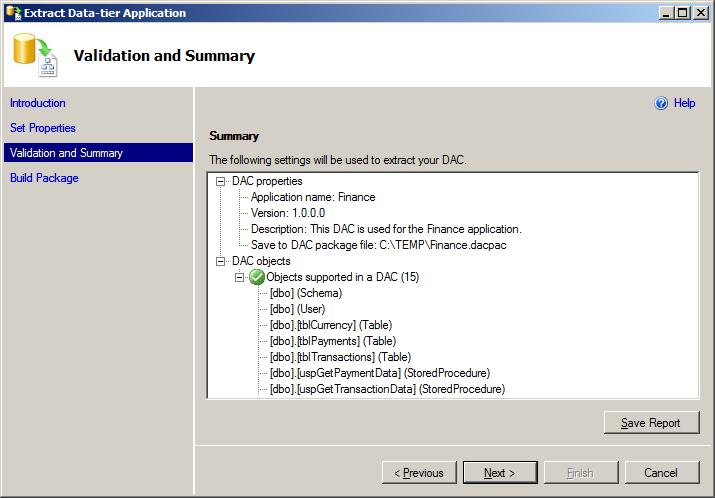
1. Register the instance of SQL Server: In the Object Explorer pane, click **Connect** and follow the instructions in the **Connect to Server** dialog box. Make sure that the login used to register the instance meets the minimal extraction [permissions](#_Minimal_permissions_and). Also, make sure that the instance you register meets the data-tier application SQL Server version [limitations](#_SQL_Server_version_1).
2. In Object Explorer, select the node of the instance from which you want to extract a data-tier application and expand its **Databases** node.
3. Select the database for the extraction. Right-click the database node, point to **Tasks**, and then click **Extract Data-tier Application** to launch the Extract Data-tier Application Wizard.
4. The first screen of the wizard is an introduction. Read the description, and then click **Next**.
5. On the **Set** **Properties** page, review or change the properties shown in figure 6 and then click **Next**. These properties are displayed in Visual Studio and in SQL Server Management Studio and are used as follows:
   * **Application Name** — Identifies the application. For example, if a database called FinanceDB serves the Finance application, the application name should be set to *Finance*. The application name is used when a data-tier application is [upgraded](#_Upgrading_a_data-tier): In order to upgrade a DAC V1 to a DAC V2, the application names of V1 and V2 must be identical.
   * **Version** — The version of the data-tier application. By default, the version number is 1.0.0.0.
   * **Description** - Optional.
   * **Save to DAC package file** — The file name and path for the DAC package file.This file must end with the *.dacpac* extension.



**Figure 6:** Setting the DAC package properties in the Extract Data-tier Application Wizard

1. When the Validation and Summary page initiates, the wizard retrieves the database objects and relevant logins and checks that a DAC package can be created for the database. The validation process ensures that all database objects are [supported](#_Objects_supported_in) by a DAC, and that all dependent objects are available in the database. For example, if a view depends on a table and the latter was dropped from the database, a DAC cannot be extracted.

When the validation completes, all issues and errors that prevent the creation of a DAC package appear in the summary tree. If there are no issues (as in figure 7), review the validation report and then click **Next** to create and save the DAC package file.



**Figure 7:** Reviewing the validation results and summary in the Extract Data-tier Application Wizard

1. On the **Build Package** page, review the status of building the DAC package file. When the build is finished, click **Finish** to close the wizard. Finally, verify that a new DAC package file was created in the path you specified in step 5.

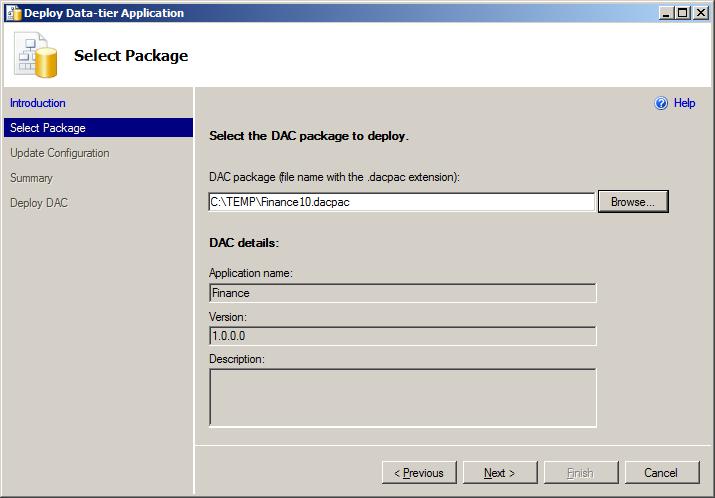
The extracted DAC package file can now be [imported](#_Creating_and_initializing) to a Data-tier Application project in Visual Studio. In addition, the DAC package can be [deployed](#_Installing_a_new) to create a new data-tier application on an instance of SQL Server, or to [upgrade](#_Upgrading_a_data-tier) an older version of an existing data-tier application.

## Installing a new DAC instance with the Deploy Data-tier Application Wizard

The Deploy Data-tier Application Wizard enables users to install a DAC package to create a new database for an application. It is recommended that the wizard be used when an application is installed for the first time. If the data-tier application was installed in the past, the [Upgrade Data-tier Application Wizard](#_Upgrading_a_data-tier) should be used instead, to push changes from a DAC package to the existing database.

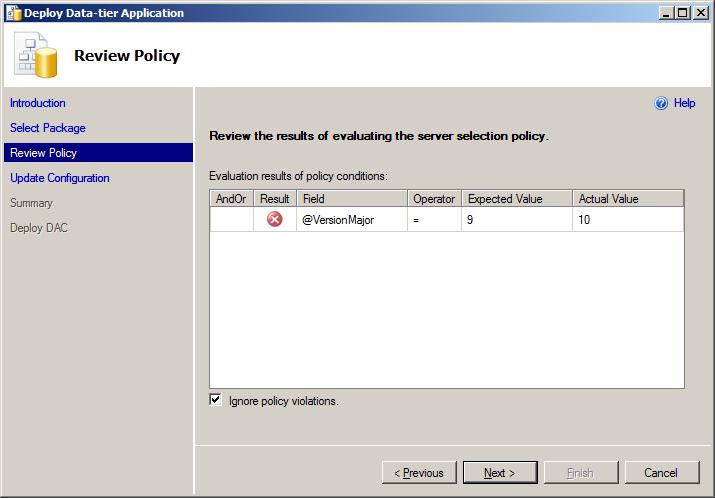
To deploy a new data-tier application, follow these instructions:

1. Locate the DAC package file that you want to deploy.
2. In SQL Server Management Studio, in the Object Explorer pane, select the node of the target instance of SQL Server on which you want to deploy the new data-tier application and create a new database. Make sure that the instance meets the data-tier application SQL Server version [limitations](#_SQL_Server_version_1) and that the instance is registered in Object Explorer under a login that meets the minimal required [credentials](#_Minimal_permissions_and). After you select the instance, expand the **Management** node, right-click the **Data-tier Applications** node, and then click **Deploy** **Data-tier Application** to launch the wizard.
3. The first screen of the wizard is an introduction. Read the description, and then click **Next**.
4. On the **Select Package** page, click **Browse** and navigate to the file you located in step 1. Examine and review the DAC details for the package you selected (see figure 8), and then click **Next**.



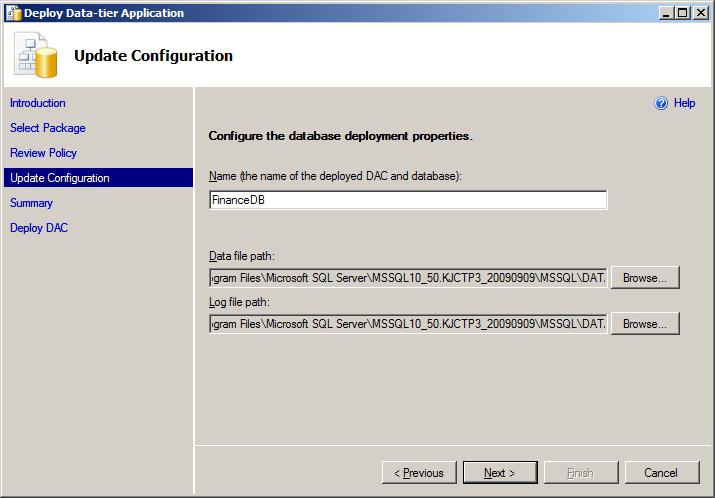
**Figure 8:** Selecting the DAC package in the Deploy Data-tier Application Wizard

1. If the DAC package contains a server selection policy, the **Review Policy** page appears. The server selection policy is authored in Visual Studio to ensure that the target instance of SQL Server meets certain deployment requirements, which are defined by developers. For example, the selected DAC package in figure 9 can only be deployed to instances of SQL Server 2005. If all policy conditions are met, click **Next**. Otherwise, you can override the policy evaluation results by selecting the **Ignore policy violations** check box and then clicking **Next**.



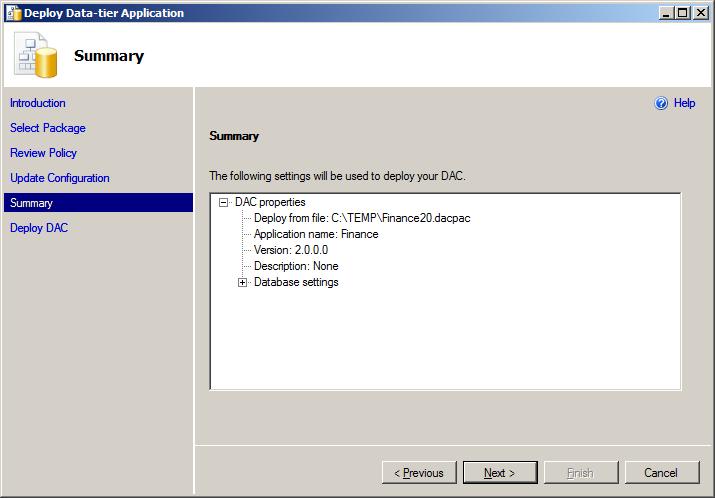
**Figure 9:** Evaluation results of the server selection policy

1. As part of the deployment, the wizard will create a new database and then populate the database and the SQL Server instance with the necessary objects. In the **Update Configuration** page, the wizard displays the database name and the path of the database files where the new database will be created. Review the database configuration options, update them if needed, and then click **Next**.



**Figure 10:** Updating the database configuration before deploying the data-tier application

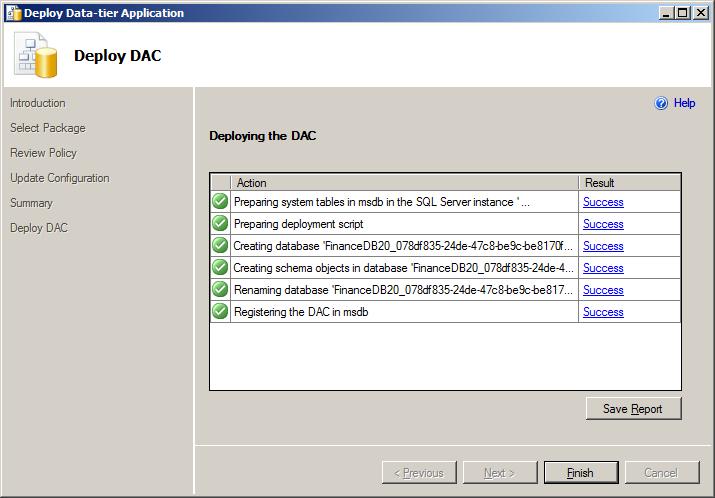
1. On the **Summary** page, review the report on the actions that will be taken to deploy the data-tier application, and then click **Next**.



**Figure 11:** Reviewing the Summary page in the Deploy Data-tier Application Wizard

1. On the **Deploy** **DAC** page, review the status of the series of actions taken to deploy the data-tier application. The wizard performs the following steps:
2. Marks the deployment in **msdb** and prepares the deployment scripts in memory.
3. Creates a new database. The database name contains a random GUID string.
4. Creates the database objects and logins.
5. Changes the random database name to the name assigned in step 7.
6. Updates **msdb** to record the DAC definition and complete the deployment.

If the deployment process encounters errors, all changes are automatically undone and rolled back by the wizard. When deployment completes, click **Finish** to close the wizard.



**Figure 12:** Final steps in the Deploy Data-tier Application Wizard

1. In the Object Explorer pane, expand the node of the instance you selected in step 2, and then refresh and expand its **Databases** node. Browse the new database, which contains the database objects deployed from the DAC package. Also, under the **Management** node, select and refresh the **Data-tier Applications** node and then expand it to view the entry for the new data-tier application.

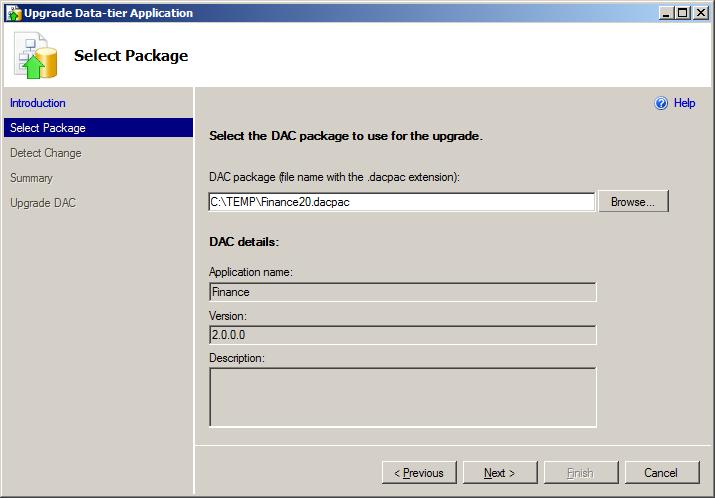
## Upgrading a data-tier application to a newer version using the wizard

The Upgrade Data-tier Application Wizard enables users to update an existing version of a DAC instance to a newer version. For example, if a DAC instance for the Finance application with version 1.0 is installed in production, and a DAC package containing version 2.0 is provided by developers, the Upgrade Data-tier Application Wizard should be run to apply the changes and push the newer version.

Before upgrading a data-tier application, it is recommended that you [compare](#_Comparing_a_DAC_1) the DAC 2.0 package content to the DAC 1.0 database to understand the differences and changes. After the comparison, take the following steps to upgrade the data-tier application:

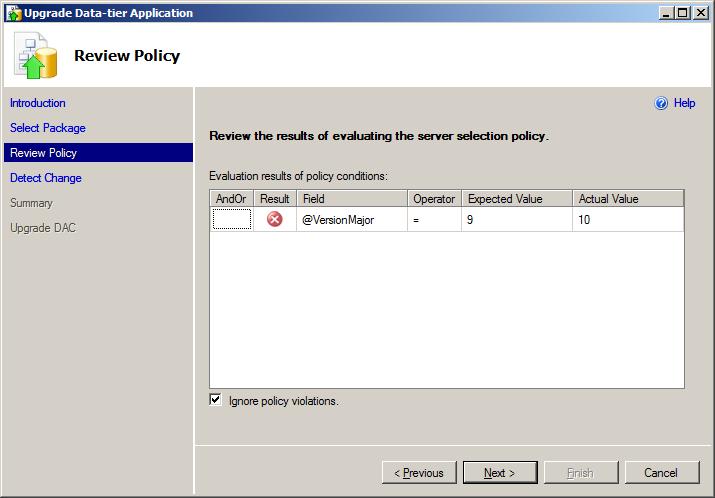
* 1. Locate the package file containing the newer version of the data-tier application.
  2. In SQL Server Management Studio, in the Object Explorer pane, select the instance of SQL Server where the current data-tier application is deployed. Make sure that the login used to register the instance in Object Explorer has sufficient [permissions](#_Minimal_permissions_and) to upgrade a data-tier application. Under the instance node, expand the **Management** node and then expand the **Data-tier Applications** node. Right-click the data-tier application to upgrade, and then click **Upgrade Data-tier Application** to launch the wizard.
  3. The first screen of the wizard is an introduction. Read the description, and then click **Next**.
  4. On the **Select Package** page, click **Browse** and navigate to the file from step 1. Review the DAC details for the package you selected (see figure 13), and then click **Next**.

In order to upgrade a data-tier application, the a*pplication name* in the DAC package must match the application name of the data-tier application on the SQL Server instance. The application name for the DAC package can be [set in Visual Studio](#_Configuring_a_data-tier). In SQL Server Management Studio, the application name of an existing data-tier application can be viewed in the Object Explorer Details (see figure 3). Users can [change the application name](#_Changing_the_application) in SQL Server Management Studio through some manual steps, although it is not recommended.



**Figure 13:** Selecting a DAC package with a newer version in the Upgrade Data-tier Application Wizard

* 1. If the DAC package contains a server selection policy, the **Review Policy** page appears next. This policy is [composed](#_Creating_a_server) in Visual Studio by developers, to check that the target instance of SQL Server meets the deployment requirements of the application. To illustrate the server selection policy, figure 14 shows a DAC package that can only be deployed to instances of SQL Server 2005. If the policy conditions are met, click **Next**. Otherwise you can override the evaluation results by selecting the **Ignore policy violations** check box and then clicking **Next**.

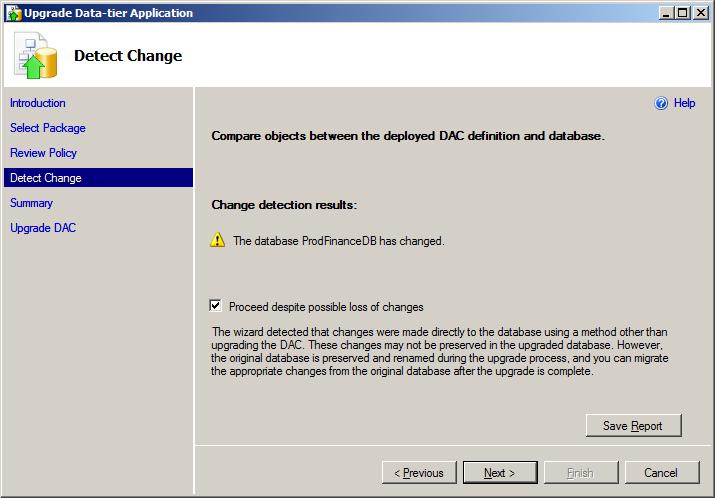


**Figure 14:** Reviewing the evaluation results of the server selection policy

* 1. SQL Server Management Studio takes the steps illustrated in figure 4 to upgrade data-tier applications in a side-by-side fashion. These steps are explained in detail later in this section. Following the ongoing example, during the upgrade, a new database is created to hold the objects specified in the DAC 2.0 package file. The name of the new database contains a random string. Then, the original database used by DAC 1.0 is set to read-only mode, and data is copied from the original database to the newly created database. Finally, the original database is renamed, the new database takes the name of the original DAC 1.0 database, and the DAC definition is updated in **msdb**.

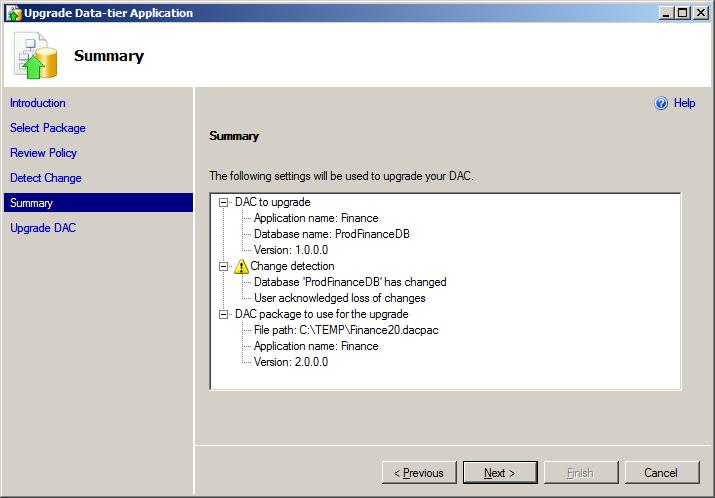
The upgrade process is called *side-by-side* because it creates a new database to hold the DAC 2.0 objects, which coexists with the DAC 1.0 database on the same instance of SQL Server. In this process, it is important to note that if any changes were made to the DAC 1.0 database outside the data-tier application framework, those changes will be lost during the upgrade. Examples of such changes include running SQL scripts manually to create objects in the DAC 1.0 database (instead of using Visual Studio to make schema changes, building a new DAC package, and upgrading the data-tier application in SQL Server Management Studio).

In order to identify the objects that may be lost during the upgrade, the **Detect Change** page in the wizard compares the DAC 1.0 definition in **msdb** to the database used by DAC 1.0. If any changes are found, the wizard alerts you and prompts you to acknowledge the possible loss of objects. In addition, you can save the list of modified objects to an html report by clicking **Save Report**. When you are done, click **Next**.



**Figure 15:** Comparing changes between the DAC 1.0 definition and database

* 1. On the **Summary** page, review the report on the actions that will be taken to upgrade the DAC, and then click **Next**.



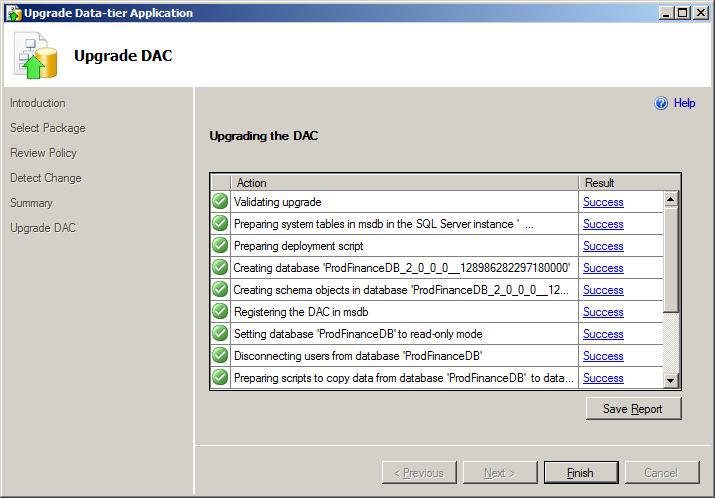
**Figure 16:** Viewing the summary report in the Upgrade Data-tier Application Wizard

* 1. On the **Upgrade DAC** page, review the status of the steps taken to upgrade the data-tier application. The wizard performs the following steps:

1. Checks user permissions and makes sure that the application name matches between DAC 1.0 and the DAC 2.0 package file.
2. Marks the beginning of the upgrade process in **msdb**.
3. Creates the deployment script that creates a new database for DAC 2.0, together with the database objects and relevant logins.
4. Creates a new database for DAC 2.0. The database name contains a random string.
5. Installs the objects from the DAC 2.0 package in the new database and create all logins that do not already exist on the instance.
6. Updates **msdb** to denote the deployment of DAC 2.0 and store the DAC 2.0 definition.
7. Sets the database used by DAC 1.0 to read-only mode and terminates user connections to it.
8. Generates a script to copy data to the new database by performing a three-way comparison:
   1. Compares the DAC 1.0 definition from **msdb** to the original database used by DAC 1.0, and compares it to the new database containing the DAC 2.0 objects.
   2. For each table and column that exists in all three sources, generates an INSERT statement to populate tables and columns in the DAC 2.0 database from the corresponding tables and columns in the original DAC 1.0 database.
9. Disables table constraints in the DAC 2.0 database.
10. Executes the script containing all INSERT statements from the previous steps.
11. Enables the constraints that were disabled before.
12. Sets the original DAC 1.0 database to read/write mode.
13. Renames the original DAC 1.0 database. The new name contains the DAC version and a random string.
14. Renames the new DAC 2.0 database to the original database name used by DAC 1.0.
15. Updates **msdb** to record the upgrade completion.

If any errors occur during the upgrade, the entire process is rolled back automatically. When the upgrade completes, click **Finish** to close the wizard.

After upgrade, the renamed DAC 1.0 database remains on the instance. This way, data or changes in the DAC 1.0 database are not lost and can be manually migrated to the DAC 2.0 database. In addition, post-deployment scripts can be manually executed against the renamed database to transform or otherwise manipulate the original database and data as needed.



**Figure 17:** Executing the steps to complete the data-tier application upgrade

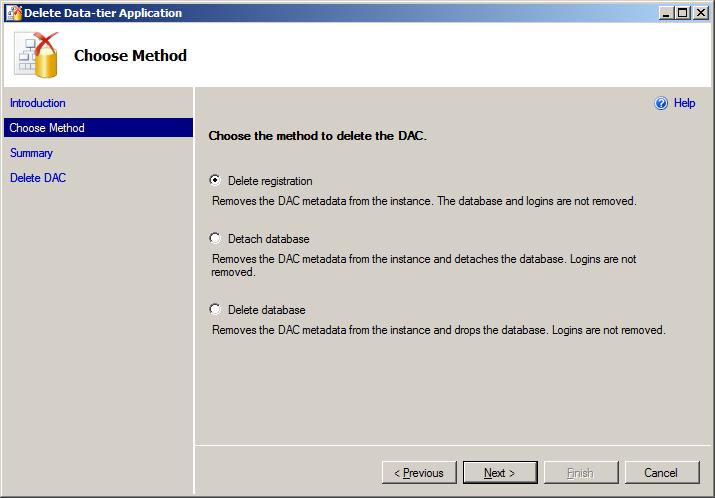
* 1. In the Object Explorer pane, expand the node of the instance you selected in step 2, and then refresh and expand its **Databases** node. Navigate and browse the DAC 2.0 and DAC 1.0 databases. Also, under the **Management** node, select and refresh the **Data-tier Applications** node. Then, select the upgraded data-tier application and refer to the Object Explorer Details pane to verify the new version.

## Deleting a data-tier application from an instance of SQL Server

A data-tier application can be removed from an instance of SQL Server using the Delete Data-tier Application Wizard or through [PowerShell](#_Deleting_one_or). To use the wizard, perform the following steps in SQL Server Management Studio:

1. In Object Explorer, locate the instance of SQL Server that contains the data-tier application you wish to remove. Make sure that the login used to register the instance has sufficient [permissions](#_Minimal_permissions_and) to delete a data-tier application. Under the instance node, expand the **Management** node and the **Data-tier Applications** node under it. Select the data-tier application to remove, right-click it, and then click **Delete Data-tier Application.**
2. The first wizard screen is an introduction that describes the wizard. Read it, and then click **Next**.
3. On the **Choose Method** page, select one of three techniques to delete the data-tier application:

* **Delete registration**— Removes the DAC definition from **msdb** without making any changes to the database or logins. After the DAC definition is removed, the data-tier application no longer exists on the instance and it does not appears in SQL Server Management Studio.
* **Detach database**— Detaches the database from the instance and then removes the DAC definition from **msdb**. Logins that are associated with the database are not changed.
* **Drop database**— Drops the database used by the data-tier application from the instance, and then removes the DAC definition from **msdb**. As before, all logins remain unchanged.



**Figure 18:** Selecting the method to remove the data-tier application from the instance

1. On the **Summary** page, review the report on the actions that will be taken to delete the DAC, and then click **Next**.
2. On the **Delete DAC** page, review the status of the series of actions taken to delete the data-tier application and then click **Finish** to close the wizard.
3. In the Object Explorer pane, click the **Data-tier Applications** node and refresh it to make sure that the data-tier application entry has been removed. If you chose the second or third options in step 3, click the **Databases** node under the instance of SQL Server and refresh it to verify that the database used by the DAC has been removed.

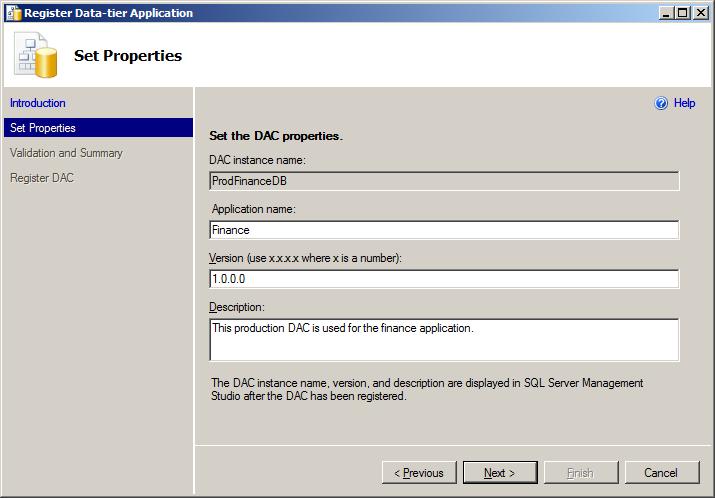
## Register a database as a data-tier application

A data-tier application consists of a database, database objects, the logins that map to the database users, and the DAC definition in **msdb**. In order to create a data-tier application in-place from an existing database, the DAC definition must be recorded in **msdb**. For this purpose, the Register Data-tier Application Wizard enables users to populate **msdb** with the necessary metadata and create a DAC for a given database.

Use the following steps to register a data-tier application in SQL Server Management Studio:

1. Refer to the Object Explorer pane and select the instance of SQL Server that holds the database to register. Make sure that the login used to connect to the instance in Object Explorer has sufficient [permissions](#_Minimal_permissions_and) to register a data-tier application. Under the instance node, expand the **Databases** node and select the database. Right-click the database node, and click **Tasks** and then click **Register as Data-tier Application** to launch the wizard.
2. The first screen of the wizard is an introduction. Read the description, and then click **Next**.
3. On the **Set** **Properties** page, review or change the properties shown in figure 19 and then click **Next**. These properties are displayed in Visual Studio and in SQL Server Management Studio and are used as follows:

* **Application Name** — Identifies the application. For example, if a database called FinanceDB serves the Finance application, the application name should be set to *Finance*. The application name is used a data-tier application is [upgraded](#_Upgrading_a_data-tier): In order to upgrade a DAC V1 to a DAC V2 version, the application name of DAC V1 and DAC V2 must be identical.
* **Version** — The version of the data-tier application. By default, the version number is 1.0.0.0.
* **Description** - Optional.



**Figure 19:** Specifying properties before registering a data-tier application

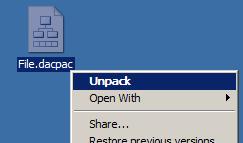
1. The **Validation and Summary** page retrieves the content of the database, checks dependencies between database objects, and ensures that all database objects are supported in a data-tier application. The validation process and the summary page content are similar to those shown for the data-tier application [extraction](#_Extract_Data-tier_Application) process. Read the summary report, and then click **Next**.
2. On the **Register DAC** page, review the status of registering the data-tier application. When registration is complete, click **Finish** to close the wizard.
3. Verify that the data-tier application is registered in Object Explorer by expanding the **Management** node for the instance and then the **Data-tier Applications** node under it. Select the new data-tier application and view its properties in the Object Explorer Details pane.

## Unpacking DAC packages to view the package content and deployment script

The *unpack* functionality is provided to let users view the content of a DAC package and generate the Transact-SQL script that corresponds to the deployment or upgrade of a data-tier application. In general, it is strongly recommended that you use the unpack feature before you deploy or upgrade a DAC package in production, to review the code and [compare](#_Comparing_a_DAC_1) or identify changes.

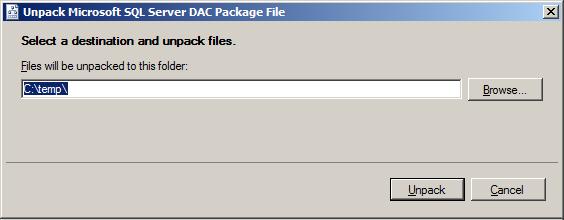
Follow these steps to unpack a DAC package:

1. Make sure that the client tools for SQL Server 2008 R2 are installed on your machine.
2. Locate a DAC package file (with the .dacpac extension).
3. Double-click the DAC package. Alternatively, you can right-click the DAC package and then click **Unpack**, as shown in figure 20.



**Figure 20:** Selecting the Unpack option from the right-click menu of a DAC package file

1. In the **Unpack Microsoft SQL Server DAC Package File** dialog box, specify a folder into which to write the DAC files.



**Figure 21:** Selecting a folder to write the DAC package content

1. Click **Unpack** to write the DAC package files and deployment script to the file system.
2. View the files. The name of the deployment script is *DACName*.sql, where DACName is the name of the data-tier application taken from the package file.

## Changing the application name or version of a deployed data-tier application

The application name for a data-tier application can be set in several locations throughout the lifecycle of a database project. The following list describes these locations and where in the lifecycle they appear:

1. Extracting a data-tier application accepts the application name as user-input, before a DAC package file is created. The application name is then embedded in the DAC package.
2. In Visual Studio, the application name can be set in the data-tier application project settings.
3. When a data-tier application is deployed, the application name is taken from the DAC package file and is recorded in **msdb**.
4. If a user registers a database as a data-tier application, the user must enter an application name, which is then recorded in **msdb**.

The application name can be viewed in the Object Explorer Details pane in SQL Server Management Studio, when you select the **Data-tier Applications** node in Object Explorer for a given instance of SQL Server. In addition, the application name plays a key role in the data-tier application upgrade process: A DAC package containing a newer version of a data-tier application can be used to upgrade an older version only if the application name is identical in both versions.

Changing the application name is not recommended. Instead, developers and DBAs should set the application name appropriately using the tools provided by the framework. However, if absolutely necessary, users can change the application name for a deployed (or registered) data-tier application through the following steps:

1. Identify the data-tier application that will be changed. Also, identify the database used by the data-tier application.
2. In SQL Server Management Studio, in Object Explorer, locate and select the data-tier application node from step 1, right-click it, and then click **Delete Data-tier Application**. Use the Delete Data-tier Application Wizard to [remove](#_Deleting_a_data-tier_1) the registration from **msdb**: In the **Choose Method** page of the wizard, click **Delete Registration**, and then complete the wizard.
3. In Object Explorer, locate and select the database node from step 1, right-click it, point to **Tasks** andthen click **Register as Data-tier Application**. Go through the [registration](#_Register_a_database) process of the data-tier application and enter the correct application name in the **Set Properties** page.

The same steps can be used to update the version of a previously deployed (or registered) data-tier application. However, it is important to note that each time the DAC definition is removed from **msdb**, the history of actions associated with the data-tier application is also removed from the instance. Therefore, in general, it is not recommended that you alter the application name or version.

## Comparing DAC packages to databases

The tools provided by the data-tier application framework in SQL Server 2008 R2 enable users to compare a DAC package to a database, or compare two databases or DAC packages to find differences. The comparison can be achieved using simple text differencing and comparison tools such as WinDiff or WinMerge.

**Compare a DAC package to a database**

The following steps enable users to compare a DAC package called Finance.dacpac to a database named FinanceDB.

1. [Unpack](#_Unpacking_DAC_packages) the DAC package file Finance.dacpac. Locate the deployment script Finance.sql.
2. [Extract](#_Extract_Data-tier_Application) a data-tier application from the FinanceDB database, and then create a DAC package called Finance1.dacpac.
3. Unpack the DAC package file created in step 2. Locate the unpacked SQL file.
4. Using a text-differencing tool, compare the SQL scripts created in step 1 and step 3.

Similar steps can be performed to compare two databases or DAC packages.

**Comparing two DAC packages**

To compare two DAC packages (for example, Finance.dacpac and Payroll.dacpac), follow these steps:

1. [Unpack](#_Unpacking_DAC_packages) the DAC package file Finance.dacpac. Locate the deployment script Finance.sql.
2. Unpack Payroll.dacpac and find the script Payroll.sql.
3. Use a text-differencing tool to compare the SQL scripts from steps 1 and 2.

**Compare databases**

The ongoing comparison concepts and techniques can be applied to compare the content of two databases (for example, the FinanceDB and PayrollDB databases):

1. [Extract](#_Extract_Data-tier_Application) a data-tier application from the FinanceDB database, and then create a DAC package called Finance.dacpac.
2. [Unpack](#_Unpacking_DAC_packages) the DAC package file Finance.dacpac. Locate the deployment script Finance.sql.
3. Extract a data-tier application from the PayrollDB database, and then create a DAC package called Payroll.dacpac.
4. Unpack Payroll.dacpac, and then find the script Payroll.sql.
5. Use a text-differencing tool to compare the SQL scripts obtained in step 2 and step 4.

## Move a data-tier application between instances

A data-tier application can be moved between instances of SQL Server through several manual steps. For example, a data-tier application can be moved by creating a full backup of the database used by a data-tier application, restoring the backup on another instance, and then [registering](#_Register_a_database) the restored database as a data-tier application. Alternatively, the database can be detached from the first instance and then attached to the second. This section describes how to use the backup and restore method.

The following steps illustrate how to move a data-tier application between two computers utilizing the backup and restore commands:

1. Create a full backup of the data-tier application database. This can be done by running the following Transact-SQL script against the instance of SQL Server containing the database. In this example, assume that the database name is FinanceDB and that the data and log backup files are created in the folder called C:\temp\.

BACKUP DATABASE FinanceDB

TO DISK = 'C:\temp\FinanceDB.bak'

GO

BACKUP LOG FinanceDB

TO DISK = 'C:\temp\FinanceDB.log'

GO

1. Copy the backup files to the target machine. In this example, assume that files are copied to the folder named C:\share\ on the target computer.
2. Restore the backup and transaction log in the instance of SQL Server running on the target machine. For example, use the following script to restore the database while moving the data and log files to the main DATA folder used by the instance.

RESTORE DATABASE FinanceDB

FROM DISK = 'C:\Share\FinanceDB.bak' WITH NORECOVERY,

MOVE 'FinanceDB\_Data' TO 'C:\Program Files\Microsoft SQL Server\MSSQL10\_50.MyInstanceName\MSSQL\DATA\FinanceDB.mdf',

MOVE 'FinanceDB\_log' TO 'C:\Program Files\Microsoft SQL Server\MSSQL10\_50.MyInstnceName\MSSQL\DATA\FinanceDB.ldf';

GO

RESTORE LOG FinanceDB

FROM DISK = 'C:\Share\FinanceDB.ldf'

GO

1. Make sure that all users in the restored database are appropriately mapped to the logins in the SQL Server instance.
2. Use the Register Data-tier Application Wizard to [register](#_Register_a_database) and create a data-tier application from the restored database.
3. Use the Delete Data-tier Application Wizard to [remove](#_Deleting_a_data-tier_1) the data-tier application from the first instance of SQL Server.

It is important to note that history records for a data-tier application are stored in **msdb**. Moving a data-tier application using the backup and restore technique (or the attach and detach technique) does not maintain the history records.

# Using PowerShell to work with data-tier applications

PowerShell enables users to write scripts and automate operations in the lifecycle of the data-tier application, as demonstrated in this chapter through simple examples. In addition, advanced code samples can be found on the Microsoft project hosting and open source software Web site at <http://www.codeplex.com>.

## Extracting a DAC package from an existing database

The following procedure creates a DAC package from an existing database using PowerShell:

1. In **Object Explorer**, start the SQLPS PowerShell program by right-clicking the instance of SQL Server that contains the database and then clicking **Start PowerShell**.
2. Instantiate the server object and assign the value of the current instance to it by running the following.

$srv = get-item .

1. Set the parameters for the extraction by running the following statements. The data-tier application is extracted from a database called FinanceDB, which serves the Finance application; the version is 1.0.0.0; and the DAC package is created at C:\temp\Finance.dacpac.

$dbname = "FinanceDB"

$applicationname = "Finance"

$version = "1.0.0.0"

$dacpacPath = "C:\temp\Finance.dacpac"

$description = "This DAC is used for the finance application"

1. Extract the data-tier application and create the DAC package file by running the following.

$extractionunit = New-Object Microsoft.SqlServer.Management.Dac.DacExtractionUnit($srv, $dbname, $applicationname, $version)

$extractionunit.Description = $description

$extractionunit.Extract($dacpacPath)

## Registering and creating a data-tier application for an existing database

You can register a database as a data-tier application on an instance of SQL Server through PowerShell. The following steps illustrate how to do this:

1. In **Object Explorer**, start the SQLPS PowerShell program by right-clicking the instance of SQL Server that contains the database and then clicking **Start PowerShell**.
2. Create a server object and set it to the current instance of SQL Server by running the following.

$srv = get-item .

1. Set the parameters for the data-tier application registration by running the following statements. The database name to register is FinanceDB, the application name for the data-tier application is Finance, and the version is 1.0.0.0.

$databasename = "FinanceDB"

$applicationname = "Finance"

$version = "1.0.0.0"

$description = "This is the DAC for the finance application"

1. Extract the data-tier application and create the DAC package file by running the following.

$registerunit = New-Object Microsoft.SqlServer.Management.Dac.DacExtractionUnit($srv, $databasename, $applicationname, $version)

$registerunit.Description = $description

$registerunit.Register()

## Deploying a new data-tier application

Users can execute the following instructions to deploy a data-tier application for the first time through PowerShell, from a DAC package to an instance of SQL Server. For more information about the deployment process, see the [overview](#_Deploying_a_DAC) and description of the deployment [wizard](#_Installing_a_new).

1. In **Object Explorer**, start the SQLPS PowerShell program by right-clicking the instance of SQL Server where you want to deploy the data-tier application and then clicking **Start PowerShell**.
2. Create a server object and set it to the current instance of SQL Server by running the following.

$srv = get-item .

1. Create a connection to your SQL Server instance and connect to the DAC store by running the following.

$serverconnection = New-Object Microsoft.SqlServer.Management.Common.ServerConnection

($srv.ConnectionContext.SqlConnectionObject)

$serverconnection.Connect()

$dacstore = New-Object Microsoft.SqlServer.Management.Dac.DacStore($serverconnection)

1. Load a DAC package from a file by running the next set of statements. In this example the Finance.dacpac file is loaded form the folder C:\temp\.

$dacpacPath = "C:\temp\Finance.dacpac"

$fileStream = [System.IO.File]::Open ($dacpacPath,[System.IO.FileMode]::OpenOrCreate)

$dacType = [Microsoft.SqlServer.Management.Dac.DacType]::Load($fileStream)

1. Subscribe to the data-tier application deployment events by running the following.

$dacstore.add\_DacActionStarted({Write-Host `n`nStarting at $(get-date) :: $\_.Description})

$dacstore.add\_DacActionFinished({Write-Host Completed at $(get-date) :: $\_.Description})

1. Replace the settings in the following statements with the appropriate values for your deployment and then execute the statements to deploy the data-tier application. Here, $dacName holds the name of the data-tier application (which is also the name of the database that will be created to store the data-tier application objects), $evaluateTSPolicy is a Boolean variable that instructs the deployment process whether to ignore server selection policy violations (if a policy exists and violations occur).

$dacName = "FinanceDB"

$evaluateTSPolicy = $true

$deployProperties = New-Object Microsoft.SqlServer.Management.Dac.DatabaseDeploymentProperties($serverconnection,$dacName)

$dacstore.Install($dacType, $deployProperties, $evaluateTSPolicy)

$fileStream.Close()

## Upgrade a data-tier application to a newer version

The instructions in this section describe how to upgrade an existing data-tier application (for example, DAC V1) to a newer version (for example, DAC V2) using PowerShell. For more information about the upgrade process, see the [overview](#_Upgrading_an_existing) section and the description of the Upgrade Data-tier Application [Wizard](#_Upgrading_a_data-tier).

The following example upgrades the Finance data-tier application from version V1 to version V2. Here, assume that V1 of the data-tier application was deployed earlier through the deployment steps outlines in the previous [section](#_Deploying_a_new).

1. In **Object Explorer**, start the SQLPS PowerShell program by right-clicking the instance of SQL Server that contains the existing version of the data-tier application that will be upgraded (that is, where DAC V1 is running) and then clicking **Start PowerShell**.
2. Create a server object and set it to the current instance of SQL Server by running the following.

$srv = get-item .

1. Create a connection to your SQL Server instance and connect to the DAC store by running the following.

$serverconnection = New-Object Microsoft.SqlServer.Management.Common.ServerConnection

($srv.ConnectionContext.SqlConnectionObject)

$serverconnection.Connect()

$dacstore = New-Object Microsoft.SqlServer.Management.Dac.DacStore($serverconnection)

1. Subscribe to the DAC store events by running the following :

$dacstore.add\_DacActionStarted({Write-Host `n`nStarting at $(get-date) :: $\_.Description})

$dacstore.add\_DacActionFinished({Write-Host Completed at $(get-date) :: $\_.Description})

1. Load the DAC V2 by running the next statements. In this example, the DAC V2 package file is called FinanceV2.dacpac and it is located in the folder C:\temp\.

$v2dac = "C:\temp\FinanceV2.dacpac"

$fileStream = [System.IO.File]::Open($v2dac, [System.IO.FileMode]::OpenOrCreate)

$v2type = [Microsoft.SqlServer.Management.Dac.DacType]::Load($fileStream)

1. Set the upgrade options. Here, $dacNameToUpgrade is the name of the existing data-tier application (and the name of the database) that will be upgraded, $evaluateTSPolicy is a Boolean value that instructs the upgrade process whether the results of the server selection policy should be ignored, and $ignoreDriftOption is a Boolean value that is provided to acknowledge the fact that if any changes were made to the data-tier application database outside the data-tier application framework, such changes can be lost during the upgrade process (for more information, see the Upgrade Data-tier Application [Wizard](#_Upgrading_a_data-tier) section).

$dacNameToUpgrade = "FinanceDB"

$evaluateTSPolicy = $true

$ignoreDriftOption = $true

1. Upgrade the data-tier application by running the following.

$options = New-Object Microsoft.SqlServer.Management.Dac.DacUpgradeOptions($evaluateTSPolicy, $ignoreDriftOption)

$dacstore.Upgrade($$dacNameToUpgrade, $v2type,$options)

$fileStream.Close()

## Deleting a data-tier application

The process to delete a data-tier application from an instance of SQL Server is described in detail in the [overview](#_Deleting_a_DAC) section of this document. Additional information is also available in the section describing the Delete Data-tier Application [Wizard](#_Deleting_a_data-tier_1).

The following PowerShell statements can be run to remove a data-tier application:

1. In **Object Explorer**, start the SQLPS PowerShell program by right-clicking the instance of SQL Server that contains the data-tier application that will be removed and then clicking **Start PowerShell**.
2. Create a server object and set it to the current instance of SQL Server by running the following.

$srv = get-item .

1. Initiate the connection to the instance of SQL Server and to the DAC Store and subscribe to the events by running the following.

$serverconnection = New-Object Microsoft.SqlServer.Management.Common.ServerConnection($srv.ConnectionContext.SqlConnectionObject)

$serverconnection.Connect()

$dacstore = New-Object Microsoft.SqlServer.Management.Dac.DacStore($serverconnection)

$dacstore.add\_DacActionStarted({Write-Host `n`nStarting at $(get-date) :: $\_.Description})

$dacstore.add\_DacActionFinished({Write-Host Completed at $(get-date) :: $\_.Description})

1. Set the name of the data-tier application that will be removed. In this example, the data-tier application and the database are named FinanceDB.

$dacName = "FinanceDB"

1. Remove the data-tier application by executing one of the following three sets of actions (for more information, see the Delete Data-tier Application [Wizard](#_Deleting_a_data-tier_1) section).
   * Remove the data-tier application by deleting the DAC definition from **msdb**.

$dacstore.Unmanage($dacName)

* + Remove the DAC definition from **msdb** and detach the database used by the data-tier application.

$dacstore.Uninstall($dacName, [Microsoft.SqlServer.Management.Dac.DacUninstallMode]::DetachDatabase)

* + Remove the DAC definition from **msdb** and delete the database used by the data-tier application.

$dacstore.Uninstall($dacName, [Microsoft.SqlServer.Management.Dac.DacUninstallMode]::DropDatabase)

# Developing and testing data-tier applications in Visual Studio 2010

## Introduction

Microsoft Visual Studio 2010 delivers several database projects and templates that enable users to develop the data portion of their applications, in the Visual Studio Professional, Visual Studio Premium, and Visual Studio Ultimate editions:

**Database Project**: Database project templates are available for SQL Server 2005 and SQL Server 2008. You can use these templates to define databases for mission-critical and business-critical applications. All objects implemented by the respective SQL Server version are supported in database projects.

**Server Project**: Server project templates are available for SQL Server 2005 and SQL Server 2008. You can use these templates to define server level objects as well as modifications to the SQL Server **master** database. Examples of server-level objects are logins and custom error messages.

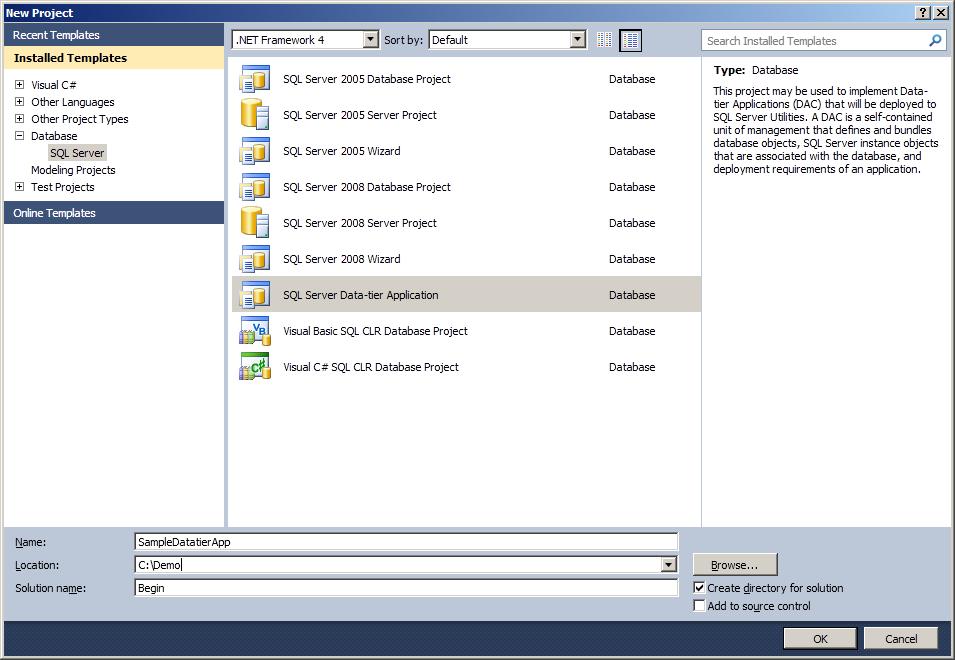
**Data-tier Application**: Data-tier application (DAC) projects are useful for implementing departmental applications (or other simple applications) that follow the [DAC support and limitations](#_Limitations_and_support_1) guidelines. With the new data-tier application project system, developers author database and server objects for their application, declare their deployment intent, and then build and package their release into a DAC package file. After it is available, the DAC package can be handed to DBAs in the test and production environments. Therefore, the DAC package operates as a single unit of deployment throughout the development, test and production lifecycle of an application.

In addition to these projects and templates, Visual Studio 2010 introduces richer development features and experiences including a new Transact-SQL editor and debugger, online and offline IntelliSense that enables users to write code faster regardless of whether the editor window is connected to an underlying database, enhanced language services, a policy designer, and Transact-SQL build services and static code analysis. For more information about these new development features, see SQL Server Books Online and online training papers.

## Creating and initializing a data-tier application

To start using data-tier applications in Visual Studio 2010, developers must first create a new data-tier application project. After the project is available, users can [author](#_Adding_tables_to) new database and server objects. Alternatively, to initialize the project with existing schema, developers can employ one of two techniques to populate the project with database and server objects for existing databases and applications. More explicitly, users can import a data-tier application from an existing database, or they can load schema objects from a DAC package file. The project creation and the steps to import objects are explained in detail here:

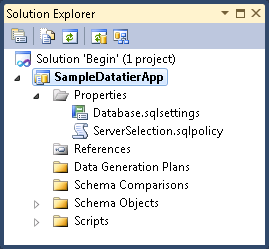
Create a new data-tier application project. In Visual Studio, click **File**, click **New**, and then click **Project**. Under **Installed Templates**, expand the **Database** node, and then click the **SQL Server** node. In the template pane, click **SQL Server Data-tier Application**. Then, type the project name, location, and solution name in the appropriate text boxes at the bottom of the form. To use source control for database code, select the **Add source control** check box. After all selections are entered, click **OK** to create the project.



**Figure 22:** Selecting the data-tier application project template

Explore the data-tier project. In the **Solution Explorer** pane in Visual Studio, find and browse through the following nodes and object types:

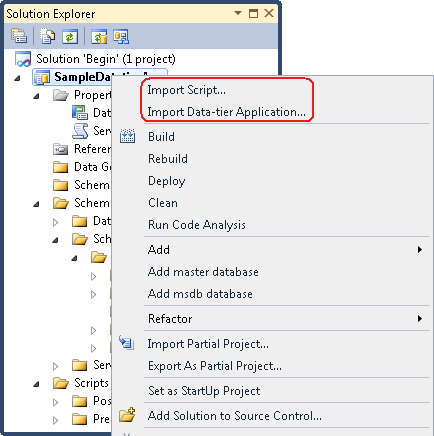
* **Properties**: contains property values that let users control how your project is deployed.
* **Data Generation Plans**: contains information about how to generate realistic and representative test data for the database and instance of SQL Server where the DAC will be deployed.
* **Schema Comparisons**: contains information about a specific comparison between your project and another schema.
* **Schema Objects**: contains database and SQL Server instance objects that are stored in SQL files. For most objects, one file is used to store one object in the project folder.
* **Scripts**: contains pre-deployment and post-deployment scripts, in addition to any scripts that you might use to manage your database or server. When you build the project, these scripts are included in the DAC package file.



**Figure 23:** Exploring the data-tier application project nodes

Initialize the project. There are several ways to populate objects and code into the project system. You can import objects from an existing database, import objects from a DAC package file, or import individual Transact-SQL scripts containing database code:

* To import objects from a database, follow the steps below. Here, it is important to note that objects can be imported from a database one time only, to initialize the project objects.
  + Open the Data-tier Application Wizard. Right-click the project node, and then click **Import Data-tier Application**.



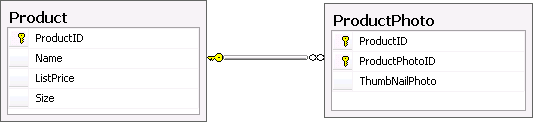
**Figure 24:** Selecting the import option to load objects to the data-tier application project

* + In the **Welcome** page of the wizard, read the instructions and then click **Next**.
  + In the **Specify Import Options** page, click **Import from an existing instance of SQL Server**. Then, type a value in **Connection string** for the SQL Server instance, or click **Edit** to configure a connection. After that, choose a database from the **Database name** list. When you are done, click **Next**.
  + The **Summary** page lists all database objects that are supported by a DAC. If any objects are not supported, the import cannot be performed successfully. If all objects are supported, click **Next**.
  + The **Import Objects** page tracks progress while the wizard creates the SQL scripts for the database objects and loads them into the project. When the page is complete, click **Finish** to close the wizard.
* Objects can be imported from a DAC package after a package file is available (for example, after the extraction of a DAC package file for an existing system in production using the Extract Data-tier Application Wizard in SQL Server Management Studio). Importing from a DAC package can be done only once, to initialize a new data-tier application project, as explained in the following instructions:
  + Start the Import Data-tier Application Wizard as illustrated in figure 24. Right-click the project node, and then click **Import Data-tier Application**.
  + Read the instructions in the **Welcome** page, and then click **Next**.
  + In the **Specify Import Options** page, click **Import from a data-tier application package** and then click **Browse** to select the DAC package file. When you are done, click **Next**.
  + The **Summary** page shows the list of database objects in the DAC package. Review the list of objects, and then click **Next**.
  + The **Import Objects** page tracks progress as the wizard creates the SQL scripts for the database objects and loads them into the project. Click **Finish** to close the wizard when the page completes.
* You can import objects from SQL scripts to initialize the data-tier application project, or you can load scripts after the project has been initialized (by loading objects from a DAC package, for example). To load objects from a script, follow these instructions:
  + Start the Import SQL Script File Wizard by right-clicking the project node and then clicking **Import Script**.
  + Read the instructions in the **Welcome** page, and then click **Next**.
  + In the **Select File** page, click **Browse** to select one or more SQL files to load into the project. When you are done, click **Next**.
  + The **Specify Options** page enables you to choose how objects should be imported. If the SQL script contains objects that are already part of the project, select the **Overwrite objects that already exist in the project** check box to overwrite the project objects. To include extended properties and object permissions from the SQL script, select the **Import extended properties** and the **Import permissions** check boxes, respectively. Also, you can use the items in the **Encoding** list to specify the encoding method that was used to create the SQL script. When you are done, click **Finish** to process the script.
  + In the **Summary** page, view the wizard actions and details log, and then click **Finish** to close the wizard.

After a data-tier application project has been created, users can add SQL Server database objects as well as instance objects (such as logins) to the project. Objects can be added by using built-in templates or by writing Transact-SQL code directly in the editor, without the use of templates. In this section, several examples of adding different objects are explored, namely adding tables, views, and stored procedures.

## Adding tables to the data-tier application project

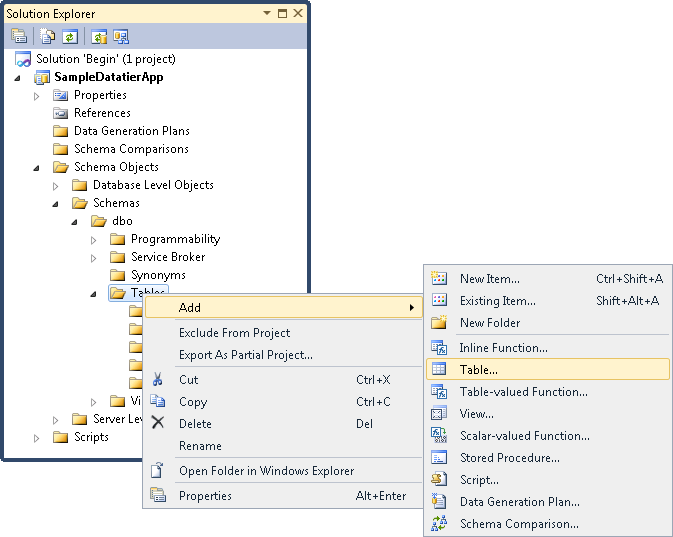
To see how tables can be added to the project system, take the following tables as an example. Here, assume that both tables belong to the dbo schema. The first table, called **Product**, has a primary key constraint on the **ProductID** column. The second table, **ProductPhoto**, has a primary key constraint on the combined value of the columns **ProductID** and **ProductPhtoID**, and the table has a foreign key relationship with the **Product** table.



**Figure 25:** Tables to add to the project system

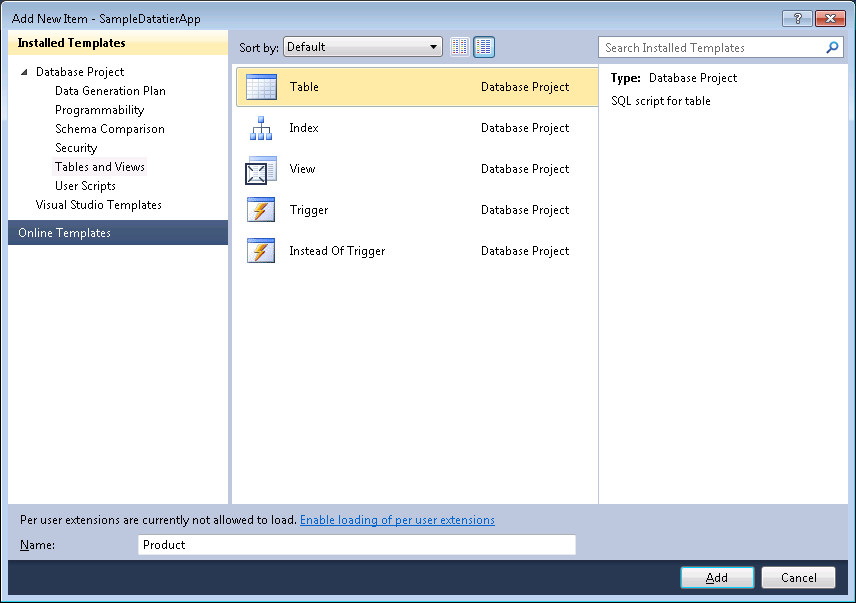
To add the tables in figure 25 to the project system, follow these instructions:

1. In **Solution Explorer**, right-click the **Tables** node located under the **Schema Objects\Schemas\dbo** folder for the project, point to **Add**, and then click **Table**.



**Figure 26:** Adding a table object to the project

1. In the **Add New Item** dialog box, set the table name to **Product**, and then click **Add**. The Product.table.sql file is added to the project and then opened in the Transact-SQL editor.



**Figure 27:** Adding a table to the project from a template

1. In the editor, replace the template code in the file Product.table.sql with the following Transact-SQL code and then save the file and the changes.

CREATE TABLE [dbo].[Product] (

[ID] [INT] IDENTITY(1,1) NOT NULL,

[Name] [NVARCHAR](50) NOT NULL,

[ListPrice] [MONEY] NOT NULL,

[Size] [NVARCHAR](5) NULL,

CONSTRAINT [PK\_Product\_ProductID] PRIMARY KEY CLUSTERED (

[ID] ASC)

1. Repeat steps 1 and 2 to create the table **ProductPhoto**.
2. Repeat step 3 for the table **ProductPhoto** and replace the template code with the following code.

CREATE TABLE [dbo].[ProductPhoto] (

[ID] [int] IDENTITY(1,1) NOT NULL,

[ProductID] [int] NOT NULL,

[ThumbNailPhoto] [varbinary](MAX) NOT NULL,

CONSTRAINT [PK\_ProductPhoto\_ProductID\_ProductPhotoID] PRIMARY KEY CLUSTERED (

[ProductID] ASC,

[ID] ASC

),

CONSTRAINT [FK\_ProductPhoto\_Product\_ProductID] FOREIGN KEY ([ProductID]) REFERENCES [Product] ([ID])

)

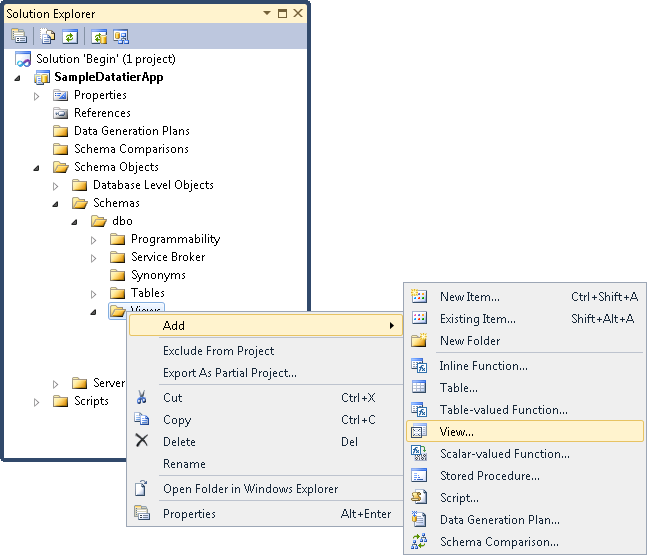
At this point, the two tables are now part of the data-tier application project. It is important to note that the CREATE TABLE statements used in this example are in accordance with the following best-practices and data-tier application [limitations](#_Limitations_and_support):

* All constraints are defined in the CREATE TABLE statement and are explicitly named in the script. In other words, the example does not use the ALTER TABLE statement and there are no system-named constraints. This fact is important because ALTER TABLE statements are not supported in the data-tier application project.
* The CREATE TABLE statement does not use the optional ON [PRIMARY] clause, which allows the user to specify the filegroup on which the table is created. The optional clause is omitted because multiple filegroups are not supported in the first release of data-tier applications in Visual Studio 2010 and SQL Server 2008 R2.

## Adding views and stored procedures to the project and building the solution

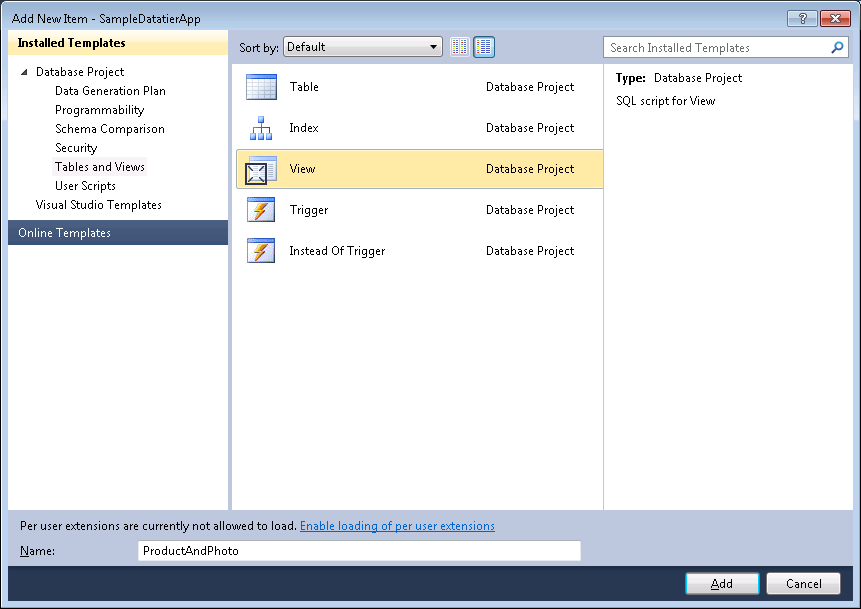
In the [previous section](#_Adding_tables_to), two tables were added to the data-tier application project. Now that the tables are available in the project system, you can create views and stored procedures that rely on these tables, by executing the following instructions. These instructions also demonstrate how to build the overall solution after all objects are added to the project.

1. Create a new view called **ProductAndPhoto** that selects from the **Products** and **ProductPhoto** tables. To do so, in **Solution Explorer,** right-click the **Views** folder located under the **Schema Objects\Schemas\dbo** folder of the project. Point to **Add**,and then click **View**.



**Figure 28:** Adding a view to the project

1. In the **Add New Item** dialog box, set the view name to **ProductAndPhoto** and then click **Add**. The ProductAndPhoto.view.sql file is added to the project and opened in the Transact-SQL Editor.



**Figure 29:** Setting the view name

1. In the editor, replace the code in the ProductPhoto.view.sql file with the following code and then save the changes.

CREATE VIEW [dbo].[ProductAndPhoto]

AS

SELECT product.\*,

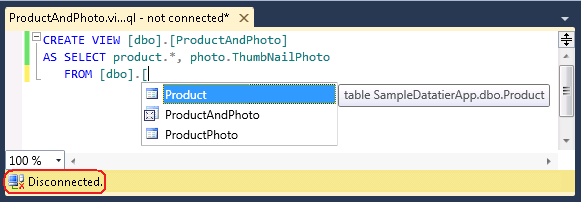
photo.[ThumbNailPhoto]

FROM [dbo].[Product]AS product

JOIN [dbo].[ProductPhoto] AS photo

ON product.[ID] = photo.[ProductID]

While writing the code in the editor, notice that IntelliSense enables you to type the code quickly even though the window is not connected to a database. This is a new feature in Visual Studio 2010 called *offline IntelliSense* (or *disconnected IntelliSense*). Here, the items listed in the completion list are those that appear in the data-tier application project scripts and files.



**Figure 30:** Offline IntelliSense features in Visual Studio 2010

1. Create a new stored procedure called **RetrieveProducts** thatgets the products from the table **Product** and **ProductPhoto**: In Solution Explorer, right-click the **Stored Procedures** folder located under the **Schema Objects\Schemas\dbo\Programmability** folder, point to **Add**, and then click **Stored Procedure**.
2. In the **Add New Item** dialog box, set the stored procedure name to **RetrieveProducts**, and then click **Add**. The RetrieveProducts.proc.sql file is added to the project and opened in the Transact-SQL editor.
3. In the editor, replace the stored procedure code with the following code and save the changes.

CREATE PROCEDURE [dbo].[RetrieveProducts]

@param1 int = 3300

AS

SELECT \*

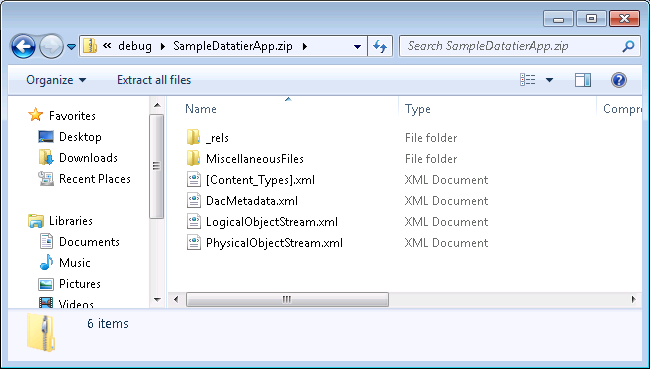
FROM [dbo].[Product]

WHERE [ListPrice] >= 3300

ORDER BY [ID] DESC

RETURN 0

1. Press CTRL+SHIFT+B to build the solution and pay attention to the Output window. In the Output window, review the output log to verify that a DAC package file has been created. In this example, the DAC package is stored as the file SampleDatatierApp.dacpac.
2. View the content of the DAC package file by navigating to the output folder and changing the extension of the SampleDatatierApp.dacpac file to .zip. Then, double-click the renamed file. A DAC package file contains XML files with a manifest that defines all the database and SQL Server instance objects used by the application. Alternatively, if the client tools for SQL Server 2008 R2 are installed on the computer, use the [Unpack functionality](#_Unpacking_DAC_packages) in order to view the content of the DAC and review the DAC deployment code.



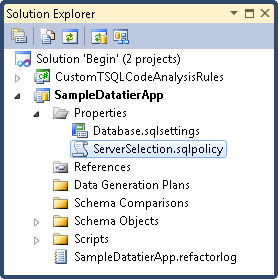
**Figure 31:** Viewing the unzipped content of the DAC package

## Creating a server selection policy to declare deployment intent

A server selection policy enables developers to set prerequisites and conditions on the target instance of SQL Server where the data-tier application is deployed. For example, server selection policies enable developers to restrict the data-tier application deployment to instances of SQL Server 2005 or later versions.

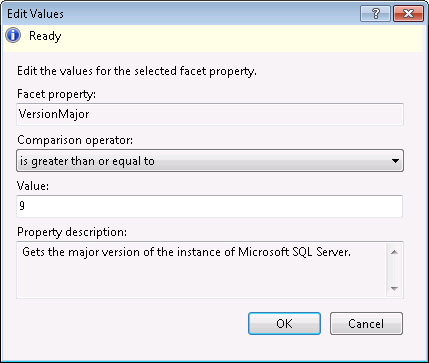
Server selection policies are composed during the [authoring of data-tier application projects](#_Authoring_data-tier_applications) in Visual Studio, and they are evaluated during [project deployment](#_Deploying_a_DAC) or [upgrade](#_Upgrade_a_data-tier). To create a server selection policy in the data-tier application project, follow these steps:

In Solution Explorer, under the **Properties** node, double-click **ServerSelection.sqlpolicy**.



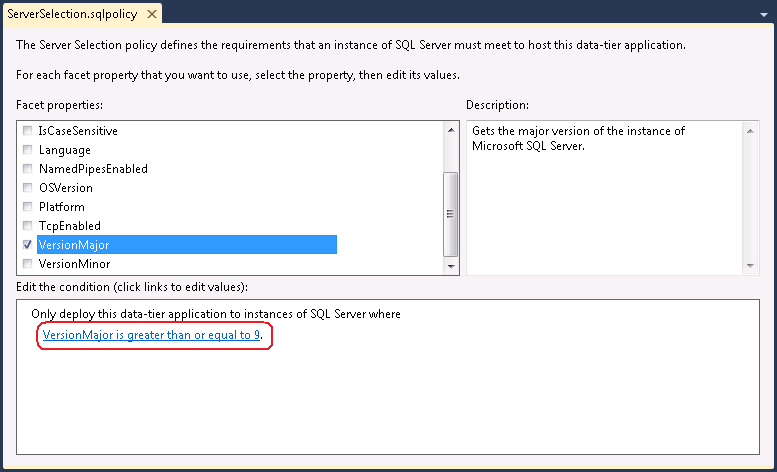
**Figure 32:** Creating a server selection policy

**Facet properties** lists the policy facets that can be defined as part of the policy conditions. For the ongoing example, select the **VersionMajor** option and select the check box next to it to open the **Edit Values** dialog box. In the dialog box, set **Comparison operator** to **is greater or equal to**, and then set the **Value** field to **9**, as illustrated in figure 33. Then, click **OK**. When you perform this action, the deployment of the data-tier application is restricted to SQL Server 2005 or later versions.



**Figure 33:** Setting a policy condition on the facet property VersionMajor

View the ServerSelection.sqlpolicy file in the editor. To save the changes, click **Save All** in the toolbar or press CTRL+SHIFT+S.



**Figure 34:** Viewing the server selection policy

## Adding a post-deployment script to the project

You can specify actions that will take place after a data-tier application is deployed to an instance of SQL Server. You specify such post-deployment actions in the post-deployment script that resides under **Scripts**, under the **Post-Deployment** folder in Solution Explorer.

To include Transact-SQL code in a post-deployment script, double-click the Script.PostDeployment.sql file and add code to the script using the editor. After changes are made, save the script to include the changes. Then, press F6 to build the solution.

When working with post-deployment scripts, you should take several considerations and best-practices into account:

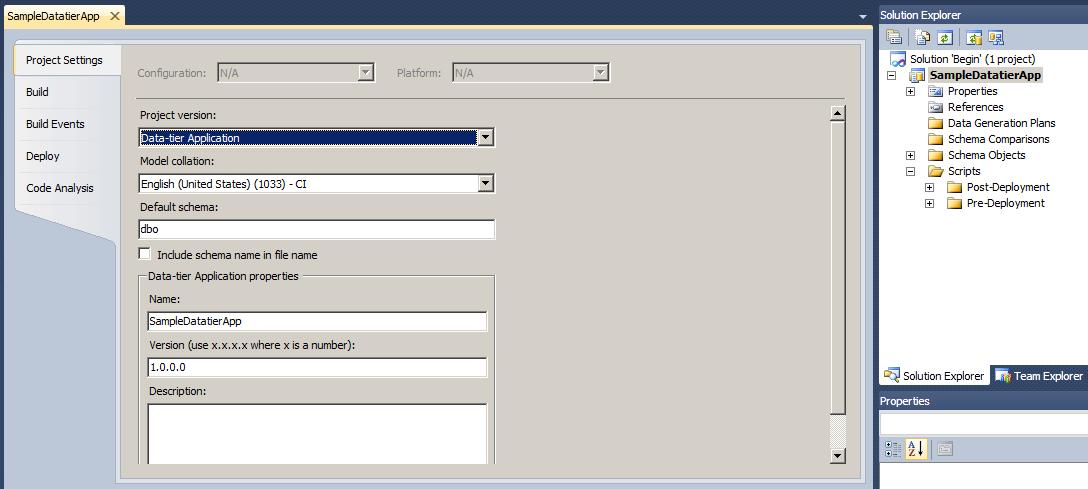
* In the first release of data-tier applications, the post-deployment script can be used to manipulate (for example, populate) data after project deployment. In addition, the post-deployment script can be used to create objects that may not be [supported](#_Limitations_and_support_1) in the initial release of data-tier applications.
* When you build the solution to create a DAC package, the post-deployment script Script.PostDeployment.sqlbecomes part of the DAC package.
* When you work in Solution Explorer, it is recommended that you include all post-deployment commands in the Script.PostDeployment.sqlscript file. This is because only one post-deployment file is included in the DAC package. In other words, you should not create multiple files.
* If you want to view the post-deployment script in the DAC package, it is recommended that you use the [Unpack Data-tier Application](#_Unpacking_DAC_packages) functionality, which is available on computers running the client tools for SQL Server 2008 R2.
* When a data-tier application is deployed from Visual Studio, the post-deployment script is automatically executed against the master database on the target instance of SQL Server. However, when deploying a data-tier application from SQL Server Management Studio, the post-deployment script is not executed automatically against the target instance. In order to apply the script changes to the data-tier application, you must run and manually execute the script against the target instance and database used by the data-tier application.
* Visual Studio identifies a post-deployment script through the **Build Action** property associated with the script. To see the property, select the script in Solution Explorer and press F4 to set focus on the Properties window, and then find and review the **Build Action** property. For a post-deployment script, the property value should be **PostDeploy**.

## Configuring a data-tier application project for deployment

Before deploying a new data-tier application to a target instance of SQL Server, or before upgrading a previously deployed data-tier application to a newer version from Visual Studio, you must first configure the project settings. The project configuration is required in order to provide Visual Studio with the connection information to the target instance of SQL Server, to set the name of the database that will be used by the data-tier application, and to set the application name.

In order to configure a data-tier application project, follow these steps:

1. Right-click the project node, and then click **Properties**. The main properties page with several property tabs is opened in the editor area.



**Figure 35:** Viewing the property tabs for the project

1. In the **Project Settings** tab, under **Data-tier Application properties**, enter the name of the data-tier application in the **Name** field. This value represents the name of your application. For example, if the application is used to handle financial data, you can set the name to *Finance*. The name of the data-tier application plays a key role during [upgrades](#_Upgrading_an_existing): If a previously deployed data-tier application that has version 1.0 is upgraded to version 2.0 using a DAC package containing the newer version, in order to perform the upgrade, the name of the data-tier applications must match. In other words, the application name stored in the DAC 2.0 package file must be identical to the name stored in the metadata tables in **msdb** for the DAC 1.0 version. For more information about the upgrade process, see the relevant section describing the [upgrade operation in SQL Server Management Studio](#_Unpacking_DAC_packages). Furthermore, it is important to note that the name of a deployed data-tier application can be [changed](#_Changing_the_application_1); however, such changes are not recommended.

In the **Project Settings** tab, you can also update the **Version** field to enter the data-tier application version for tracking and development purposes. In addition, it is recommended that you enter a short description for the project in the **Description** field. When you are done, press CTRL+S to save the updated values.

1. Click the **Build** tab. Ensure that **Database collation** is set to the appropriate value, or change it as necessary.
2. Click the **Deploy** tab and refer to **Destination connection string**. Click **Edit** to open the **Connect to server** dialog box and select the name of the instance of SQL Server where the data-tier application will be deployed. In this dialog box, make sure to use the login that you will use to deploy or upgrade the data-tier application (for more information about minimal login permissions, see the [Minimal permissions and security notes](#_Minimal_permissions_and) section). Furthermore, make sure that the version of the SQL Server instance meets the [data-tier application limitations](#_SQL_Server_version_1).



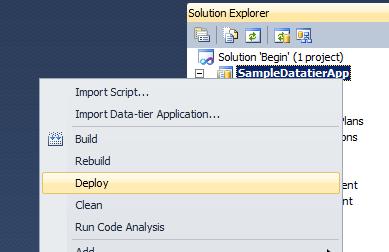
**Figure 36:** The connection dialog box

1. In the **Deploy** tab, enter the name of the database that will be used in the **Data-tier Application instance name** box.When the data-tier application is deployed, the project objects will be created in this database.
2. To ensure that the target instance meets the prerequisites specified in the [server selection policy](#_Creating_a_server), select the **Validate server selection policy** check box. Note that this option does not affect SQL Server Management Studio, where the server selection policy is always evaluated against the target instance before deployment or upgrade.

After the project is configured, if the data-tier application has never been deployed, you can use Visual Studio to [deploy](#_Deploying_a_DAC) the project and create the new database and objects. If the data-tier application was previously deployed, you can use Visual Studio to [upgrade](#_Upgrading_an_existing) the existing one to include the objects specified in the project system.

## Deploying a data-tier application

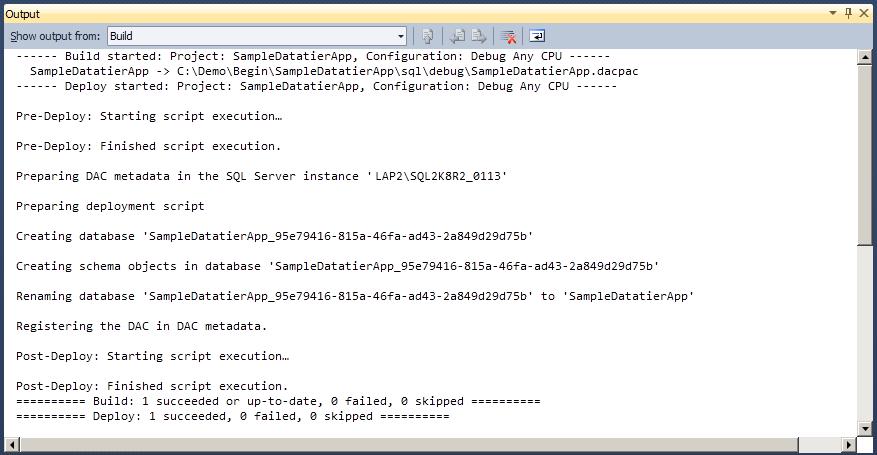
After a project is [configured](#_Configuring_a_data-tier) for deployment, you can view the output. Click **View**,and then either click **Output** or press CTRL+W to show the Output pane. Then, right-click the project node in Solution Explorer and then click **Deploy** to apply the data-tier application changes, as illustrated in figure 37.



**Figure 37:** Deploying the data-tier application

If this is the first time that the data-tier application has been deployed (that is, there is no database with the name entered in the **Deploy** properties tab for the project), Visual Studio creates a new database with the specified name and installs the database and instance objects. However, if the database already exists on the target instance and the data-tier application is being redeployed, Visual Studio upgrades the existing data-tier application. The [first-time-deployment](#_Deploying_a_DAC) and [data-tier application upgrade](#_Upgrading_an_existing) are described in detail earlier in the document.

In the Output window, observe that before deployment initiates, Visual Studio builds the project and creates a DAC package file with the extension .dacpac. Then, the DAC package is deployed against the target instance. Figure 28 shows the trace log reported in the output window for a successful sample deployment. To check the deployment, use Visual Studio or SQL Server Management Studio to log in to the target instance of SQL Server and explore the database associated with the data-tier application.



**Figure 38:** The output log for a successful deployment

In several scenarios, Visual Studio deployments will result in errors. For example, such events occur if the conditions in the [server selection policy](#_Creating_a_server) are not satisfied by the target instance of SQL Server, if the project is not [configured](#_Configuring_a_data-tier) properly, if Transact-SQL errors are found in the project code, or if the project contains objects that are [not supported](#_Limitations_and_support_1) by a data-tier application. If deployment errors occur, it is strongly recommended that you refer to the Output window and view the error log.

## Upgrading a data-tier application

When you are [deploying a data-tier application project in Visual Studio](#_Deploying_a_data-tier_1), if the data-tier application was previously deployed on the target instance, Visual Studio performs the same steps taken by the data-tier application upgrade process. This process and its associated actions are described in detail [earlier](#_Upgrading_an_existing) in the document, and in the data-tier application [upgrade section](#_Upgrading_a_data-tier) for SQL Server Management Studio.

# Limitations and support

## Objects supported in a data-tier application

The initial release of data-tier applications is designed to address simple information-technology (IT) systems. Consequently, several restrictions are imposed by the data-tier application framework.

Data-tier applications and DAC packages are designed to serve user-databases. DACs cannot be operated to model and work with system databases. For example, users cannot extract, register or deploy DAC packages for the master or msdb system databases.

The list of database object-types supported in a DAC is available in SQL Server Books Online at <http://msdn.microsoft.com/en-us/library/ee210549(SQL.105).aspx>. Common database objects that are not supported in a DAC include:

* Objects marked for deprecation, including defaults, rules and numbered stored procedures
* CLR objects and data types (such as Spatial, Geography, Geometry, Hierarchy ID data types, SQL assemblies, CLR stored procedures and functions)
* User-defined aggregates and user-defined (CLR) types
* Partition schemes and partition functions
* XML schema collections, XML indexes and spatial indexes
* Service broker objects
* Filestream columns
* Symmetric keys, asymmetric keys, certificates
* DDL triggers
* Application roles
* Full-text catalog objects
* Extended stored procedures
* Encrypted objects (for example, encrypted stored procedures, views, functions, and triggers)
* Objects containing cross-database dependencies and linked server references
* Extended properties
* Synonyms

Passwords are not stored in the DAC package, or in the DAC definition in **msdb**. When SQL logins are created during DAC deployment or upgrade, each SQL login is created as a disabled login with a strong random password and the MUST\_CHANGE clause. Thereafter, users must enable and change the password for each new SQL login (using the ALTER LOGIN command, for example).

Several database and SQL Server instance-level objects cannot be bundled into a DAC package (when a DAC in is built or imported in Visual Studio or when a DAC is extracted in SQL Server Management Studio). When a DAC is registered or extracted, these instance-level items are not captured in the DAC definition and are not stored in **msdb** or in the DAC package, respectively. However, these objects do not impose any other limitations or prevent the use of DACs. In other words, users can extract, register, deploy, and upgrade DACs on instances containing such items.

Database objects that fall under this category include:

* Object permissions
* Role membership (mappings between users and database roles)
* Extended properties
* Statistics
* Diagrams
* Plan guides

SQL Server instance-level objects in this category include:

* Linked servers
* SQL jobs
* Certificates
* Endpoints
* Credential objects
* User-defined policies and conditions
* Role membership (mappings between logins and server roles)

The length of the database name used to extract, register or deploy the data-tier application cannot exceed 87 characters.

Files and filegroups: The DAC framework provides limited support for database files and filegroups. DACs can be extracted from a database that has multiple filegroups, and a database with multiple filegroups can be registered as a DAC. However, filegroups are not stored in the DAC definition in the DAC package or in **msdb**, after the extraction or registration process, respectively. As a result, when a DAC is deployed, the new database created during the deployment will have one filegroup and a single file. The initial size of this new database is set to 3 MB with 1 MB unrestricted auto-growth. The log file for the database is set to an initial size of 1 MB, with 10 percent auto-growth, and it is restricted to a maximum size of 2 terabytes.

## SQL Server version limitations

Data-tier applications can be extracted from SQL Server 2000 or later versions, to create a DAC package file. However, currently, data-tier applications can only be deployed to an instance of the Database Engine of SQL Server 2008 R2. Similarly, registering a database as a data-tier application or upgrading a data-tier application is supported only in SQL Server 2008 R2. Deployment support for additional versions of SQL Server may become available in future releases.

## DAC package size limitations

A DAC package can be created using one of the following techniques:

* + In SQL Server Management Studio, choose a database and run the Extract Data-tier Application Wizard. The wizard produces a DAC package file (with the extension .dacpac) that contains the database objects and the logins associated with the database users.
  + In Visual Studio, use the Build feature to create a DAC package from the Data-tier Application project system. In this scenario, the DAC package file will include a server selection policy, as well as custom user-scripts such as pre/post deployment scripts (for example, post-deployment scripts containing INSERT statements to manipulate data during deployment or upgrade).

With either technique, the size of the *extra files* section of the DAC package that stores the [post-deployment scripts](#_Adding_a_post-deployment) for the data-tier application cannot exceed 50 MB.

## Minimal permissions and security notes

The following permissions are required in the lifecycle of a data-tier application:

* To extract a DAC, you must have VIEW DEFINITION permissions and SELECT privileges on sys.sql\_expression\_dependencies in the user database, as well as ALTER ANY LOGIN privilege on the instance of SQL Server. If you belong to the db\_owner database role and the securityadmin server role, you automatically meet these requirements.
* The ability to deploy a DAC is limited to members of the dbcreator server role that also have ALTER ANY LOGIN permissions.
* To upgrade a data-tier application from (say) V1 to V2, you must be a member of the dbcreator server role, must have ALTER ANY LOGIN permissions, and must also be a dbo of the DAC V1 database.
* Registering a data-tier application requires the same permissions as deployment.
* To delete a DAC, you must be a dbo of the database used by the DAC.
* The Unpack DAC operation uses the security context of the logged-in Windows user. The unpack functionality does not require elevated permissions. In order to successfully unpack the files contained in a DAC package to a given folder, users must have permissions to write to the target folder.

For DAC logins, there are several security-related considerations:

* Passwords for SQL logins – As noted [earlier](#_Objects_supported_in), passwords for SQL logins are not modeled in the DAC. Passwords must be set manually after new SQL logins are created during deployment or upgrade.
* Cross-domain Windows logins and deployments – Say that developers use Visual Studio to compose and deploy a data-tier application in one Windows domain. When they are done, developers create a DAC package and send it to production for deployment in a different domain. However, the name of the first domain is hardcoded for all Windows logins in the package; therefore cross-domain Windows logins cannot be deployed successfully. Such scenarios result in warnings (but not errors) and do not block or prompt a rollback of the deployment. To avoid these warnings, specify the DDL statements to create logins in a pre/post-deployment script in Visual Studio.
* Default database for DAC logins - The **master** database is the default database for all logins created during DAC deployment or upgrade.
* Login name conflicts– During deployment, if a login name contained in the DAC package already exists on the instance (possibly with different properties), the creation of the login is skipped and the existing login is not changed. The same behavior occurs when an existing version of a data-tier application is upgraded using a DAC package that contains a newer version.

# Disaster recovery and high-availability considerations

A data-tier application consists of a database, the database objects and content, and the logins associated with the users of the database. Therefore, depending on the architecture of SQL Server environments where data-tier applications are utilized, your organization may need to adjust or add steps to existing disaster recovery (DR) and high-availability (HA) solutions in order to accommodate the data-tier applications restrictions. Several adjustments and recommendations are described in this section for common HA-DR solutions.

## Backup and restore

SQL Server offers database backup and restore capabilities to recover from disaster. However, in order to plan recovery solutions for data-tier applications, you should take several additional steps. For example, say that a data-tier application is running on a SQL Server instance called Instance1, and that a second instance called Instance2 will be used to recover the data-tier application. The following considerations are recommended to ensure easy recovery from Instance1 to Instance2:

* Make sure that all logins of Instance1 are also available in Instance2.
* Make sure that the database used by the data-tier application in Instance1 is backed up periodically, and that the backup files are accessible from Instance2.
* To restore a data-tier application on Instance2, restore the data-tier application database from backup. Then, [register](#_Register_a_database) the database as a data-tier application on Instance2. During the registration process, provide the application name and DAC version that were used in Instance1.
* After registration, change the connection information for your applications (for example, the connection string in a web.config file) to access the database in Instance2.

In general, it is also recommended that you periodically back up the **msdb** database or the DAC tables in **msdb**, because the application name and DAC version required can be retrieved from the backup. Moreover, instead of registering a data-tier application, you can restore the **msdb** system database from the backup of Instace1, to automatically populate the DAC definition in **msdb** in Instance2.

## Mirroring and log shipping

Data-tier applications are operable under database mirroring. However, special actions in the lifecycle of a data-tier application can impact the mirroring process and must be considered accordingly. For the sake of discussion, say that a *source* database is used on a principal instance of SQL Server called Instance1 and that a *target* mirrored database is located on the mirror instance named Instance2.

A data-tier application can be [extracted](#_Extract_Data-tier_Application) from the source database on the principal machine. However, extracting a mirror database (that is, the database on Instance2) is not a supported scenario. Similarly, the source database on Instance1 can be [registered](#_Register_a_database) as a data-tier application, but the registration functionality is not available for the target database on Instance2.

After a new data-tier application is deployed from a DAC package (for example, by using the Deploy Data-tier Application [Wizard](#_Installing_a_new)), a new database is created to hold the DAC package objects. After the database is available, you can set up mirroring between the principal and mirror instances using standard mirroring settings.

If a database is used by a data-tier application and is the source database on a principal instance, each time the data-tier application is upgraded, several manual actions must be taken in order to successfully perform the upgrade. The manual steps are necessary because the [upgrade](#_Upgrading_a_data-tier) process is performed in a side-by-side fashion: When you upgrade an existing version (say, DAC V1) to a newer version (for example, DAC V2), a new database is created to hold the objects for DAC V2, the mirrored database of DAC V1 is renamed, and the new database takes the name of the original mirrored database. Therefore, to upgrade a data-tier application, you must remove the mirroring session. One way to end the mirroring session is to run the following on the principal instance.

ALTER DATABASE DatabaseName SET PARTNER OFF;

After mirroring is disconnected, the upgrade process can be executed. When the upgrade completes, create a backup of the database used by the data-tier application (that is, the newly created database), restore it on the mirror instance, and set up mirroring again between the principal and the mirror instances.

For deletion, data-tier applications can be removed from an instance of SQL Server in one of three ways: Removing the DAC definition from **msdb**, removing the DAC definition and detaching the database, or removing the definition and dropping the database. If the database used by the data-tier application is mirrored, the ALTER DATABASE statement above must be executed before the database can be detached or removed. However, no steps are required in order to remove the DAC definition from **msdb**, because it does not affect the source database.

Finally, it is important to note that similar concepts apply to log shipping: Extraction and registration actions are available on the source database. Also, log shipping should be disconnected before upgrading a data-tier application, and it should be set up again after the upgrade completes.

## Clustering

The various actions in the lifecycle of data-tier applications can be performed in a clustered environment without the need to perform special actions. Said differently, data-tier applications can be operated, deployed or upgraded on clustered instances without the need to perform additional or custom manual steps.

# Conclusion

With the introduction of data-tier applications in SQL Server 2008 R2 and Visual Studio 2010, Microsoft facilitates the lifecycle of database projects by enabling developers to compose database code under source control, declare and express deployment intents, package changes into a single DAC package and then transfer changes to the test and production SQL Server environments. Database administrators in the test and production environments can then automatically deploy database changes using built-in tools, which omits the need to run error-prone and potentially time consuming scripts.

# End notes

[D01] If one or more of the policy conditions are not satisfied, you can choose to ignore the policy violation and proceed, or you can abort the deployment.

[D02] The original DAC V1 database is renamed and is left on the SQL Server instance after the upgrade completes. Therefore, changes are not completely lost as they can be manually recovered and moved to the DAC V2 database.

# Feedback

Did this paper help you? Please give us your feedback. Tell us on a scale of 1 (poor) to 5 (excellent), how would you rate this paper and why have you given it this rating? For example:

* Are you rating it high due to having good examples, excellent screen shots, clear writing, or another reason?
* Are you rating it low due to poor examples, fuzzy screen shots, or unclear writing?

This feedback will help us improve the quality of white papers we release.

[Send feedback](mailto:sqlfback@microsoft.com?subject=White%20Paper%20Feedback:%20Data-tier%20Applications%20in%20SQL%20Server%202008%20R2).