[MS-DPDACPAC]:
Data-Tier Application Data Portability Overview

Revision Summary

Date	Revision History	Revision Class	Comments
06/04/2010	0.1	Major	First release.
09/03/2010	0.1.1	Editorial	Changed language and formatting in the technical content.
02/09/2011	0.1.1	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1   Introduction

A data-tier application (DAC) is a self-contained unit of deployment that enables data-tier developers and database administrators (DBAs) to package Microsoft® SQL Server® objects, including **database** and instance objects, into a single entity called a DAC package (a .dacpac file), as specified in [MSDN-UNDERDAC]. A .dacpac file consists of a package of XML parts that represents metadata of the data-tier application and SQL Server object **schema** [MS-DACPAC].

This document provides an overview of data portability scenarios, data export and import, between Microsoft® SQL Server® 2008 R2 and a vendor's application using a .dacpac file as a portable artifact. In these scenarios, a vendor must provide API or XML transformation methodology to produce or consume the .dacpac file within the vendor's application, unless it is implemented using the Microsoft DAC API [MSDN-DACAPI].
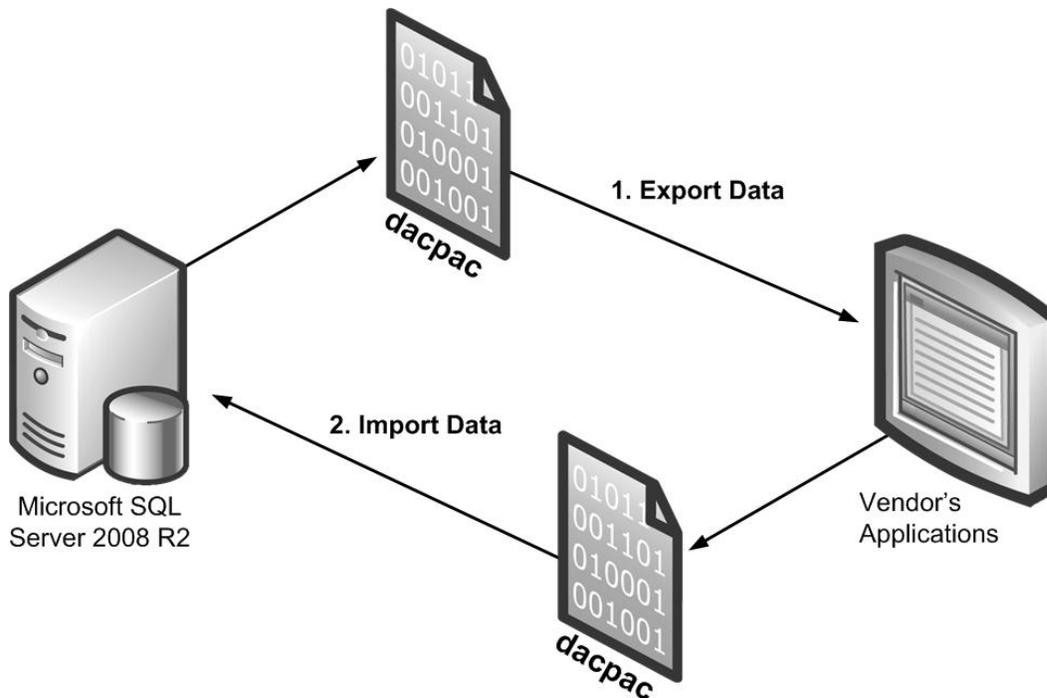
Figure 1: Conceptual overview of export and import data portability

In the (1) export data scenario in the preceding figure, a vendor can implement an application using the DAC API as specified in [MSDN-DACAPI] to export SQL Server objects to a .dacpac file. The methodology for exporting SQL Server objects to a .dacpac file is described in section 2.1.

In the (2) import data scenario in the preceding figure, a vendor can implement an application using the DAC API as specified in [MSDN-DACAPI] to import the vendor-produced .dacpac file into SQL Server 2008 R2. This methodology is described in section 2.2.

1.1   Glossary

The following terms are defined in [MS-GLOS]:

database
schema

1.2 References

[MS-DACPAC] Microsoft Corporation, "Data-Tier Application File Format Structure Specification".

[MSDN-DACAPI] Microsoft Corporation, "Microsoft.SqlServer.Management.DAC Namespace", http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.management.dac(SQL.105).aspx

[MSDN-DACERROR] Microsoft Corporation, "Troubleshooting Data-tier Applications", http://msdn.microsoft.com/en-us/library/ee240741(SQL.105).aspx

[MSDN-DACEXCON] Microsoft Corporation, "DacExtractionUnit Constructor", http://msdn.microsoft.com/en-us/library/ee211479(SQL.105).aspx

[MSDN-DACEXT] Microsoft Corporation, "DacExtractionUnit Class", http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.management.dac.dacextractionunit(SQL.105).aspx

[MSDN-DACEXUEX] Microsoft Corporation, "DacExtractionUnit.Extract Method", http://msdn.microsoft.com/en-us/library/ee211479(SQL.105).aspx

[MSDN-DACSTIN] Microsoft Corporation, "DacStore.Install Method", http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.management.dac.dacstore.install(SQL.105).aspx

[MSDN-DACSUPOB] Microsoft Corporation, "SQL Server Objects Supported in Data-tier Applications", http://msdn.microsoft.com/en-us/library/ee210549(SQL.105).aspx

[MSDN-DACTYCON] Microsoft Corporation, "DacType Constructor", http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.management.dac.dactype.dactype(SQL.105).aspx

[MSDN-DACTYPE] Microsoft Corporation, "DacType Class", http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.management.dac.dactype.dactype(SQL.105).aspx

[MSDN-DBSTATE] Microsoft Corporation, "Database States", http://msdn.microsoft.com/en-us/library/ms190442.aspx

[MSDN-PACKGET] Microsoft Corporation "Package.GetPart Method", http://msdn.microsoft.com/en-us/library/system.io.packaging.package.getpart.aspx

[MSDN-PACKNAME] Microsoft Corporation, "System.IO.Packaging Namespace", http://msdn.microsoft.com/en-us/library/system.io.packaging.aspx

[MSDN-PACKOP] Microsoft Corporation "Package.Open Method", http://msdn.microsoft.com/en-us/library/system.io.packaging.package.open.aspx

[MSDN-PACKPARTCON] Microsoft Corporation, "PackagePart Constructor", http://msdn.microsoft.com/en-us/library/system.io.packaging.packagepart.packagepart.aspx

[MSDN-UNDERDAC] Microsoft Corporation, "Understanding Data-tier Applications", http://msdn.microsoft.com/en-us/library/ee240739(SQL.105).aspx

Release: Wednesday, February 9, 2011

2 Data Portability Scenarios

2.1 Export Data

The data export scenario describes export customer data from Microsoft SQL Server 2008 R2 to a .dacpac file so that a vendor can consume it within the vendor's application. As shown in the following figure, a .dacpac file can be created by extracting SQL Server objects and then unzipped to XML parts. A vendor can consume the XML parts of a .dacpac file as a native XML format. In this case, the vendor must implement the methodology to consume the .dacpac file within the vendor's application.

As shown in the following figure, a .dacpac file consists of dacmetadata.xml, logicalobjectstream.xml, physicalobjectstream.xml. It may contain targetselection.xml and miscellaneous files, such as Transact-SQL scripts. Refer to [MS-DACPAC] for more detail of the file format structure.
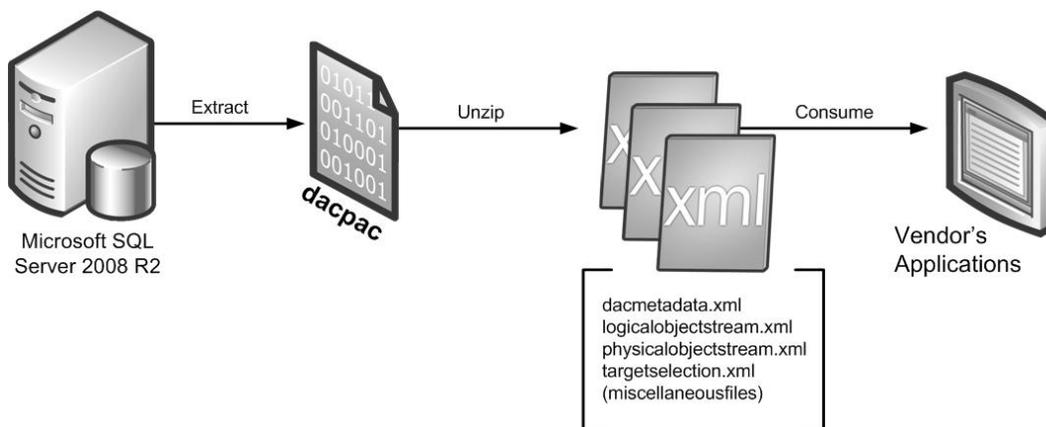
Figure 2: Export data

This section provides a step-by-step description and references for exporting data to a .dacpac file and obtaining XML parts using APIs.

2.1.1 Data Description

Customer data

The customer data is a schema representation of a SQL Server database and instances in Microsoft® SQL Server® 2008 R2. In this version, a .dacpac file supports a subset of SQL Server objects, as specified in [MSDN-DACSUPOB].

Intended user

The intended user is a vendor who can export SQL Server object schema from SQL Server 2008 R2 to a .dacpac file format to consume it within the vendor's application.

2.1.2 Format and Protocol Summary

The following table provides a comprehensive list of the formats and protocols used in the export data portability scenario.

Protocol or format name	Description	Reference
Data-Tier Application File (.dacpac file) Format	The data-tier application file format is a package of XML files that serves as the packaging format for the data-tier application.	[MS-DACPAC]
Microsoft.SqlServer.Management.DAC Namespace	The **Microsoft.SqlServer.Management.Dac** namespace contains classes that represent the DAC objects.	[MSDN-DACAPI]
System.IO.Packaging Namespace	The **System.IO.Packaging** namespace provides classes that support storage of multiple data objects in a single container.	[MSDN-PACKNAME]

2.1.3   Data Portability Methodology

The data portability methodology describes the steps to extract and unzip a data-tier application using the DAC API and **System.IO.Packaging**. The vendor's proprietary implementation for consuming the .dacpac file is outside the scope of this section.

Extract a data-tier application

To extract a data-tier application, follow these steps:

1. Initialize a new instance of the **DacType** [MSDN-DACTYPE]. For more information, refer to the **DacType** constructor [MSDN-DACTYCON].

2. Initialize a new instance of the **DacExtractionUnit** class and connect to the Microsoft® SQL Server® database [MSDN-DACEXT]. For more information, refer to the **DacExtractionUnit** constructor [MSDN-DACEXCON].

3. Extract the database to the **DacType** [MSDN-DACEXT]. For more information, refer to the **DacExtractionUnit.Extract** method [MSDN-DACEXUEX].

4. Save the **DacType** as a .dacpac file. For more information, refer to the **DacType.Save** method [MSDN-DACTYPE].

Unzip a data-tier application

To unzip a data-tier application by using **System.IO.Packaging**, follow these steps:

1. Initialize a new instance of **Package** class and open the .dacpac file [MSDN-PACKNAME]. For more information, refer to the **Package.Open** method [MSDN-PACKOP].

2. Save package parts by using a specific folder [MSDN-PACKNAME]. For more information, refer to the **Package.GetPart** method [MSDN-PACKGET].

After XML parts are created in the specified folder, a vendor's application can load it as a standard XML file for further proprietary processing.

2.1.3.1   Preconditions

The SQL Server database must be ONLINE as specified in [MSDN-DBSTATE].

2.1.3.2   Versioning

This version of the export data scenario is applicable to the Microsoft SQL Server 2008 R2 release.

Release: Wednesday, February 9, 2011

2.1.3.3 Error Handling

Data-tier application error handling and troubleshooting are described in [MSDN-DACERROR].

2.1.3.4 Coherency Requirements

The Microsoft SQL Server object must be listed as a supported object in [MSDN-DACSUPOB].

2.1.3.5 Additional Considerations

There are no additional considerations.

2.2 Import Data

The data import scenario describes importing vendor's data to a .dacpac file so that the data can be deployed to Microsoft® SQL Server® 2008 R2 as a data-tier application. As shown in the following figure, a vendor can produce XML parts that conform to [MS-DACPAC] structure format and package it to a .dacpac file. Note that the vendor must implement the methodology producing the XML parts within the vendor's application.
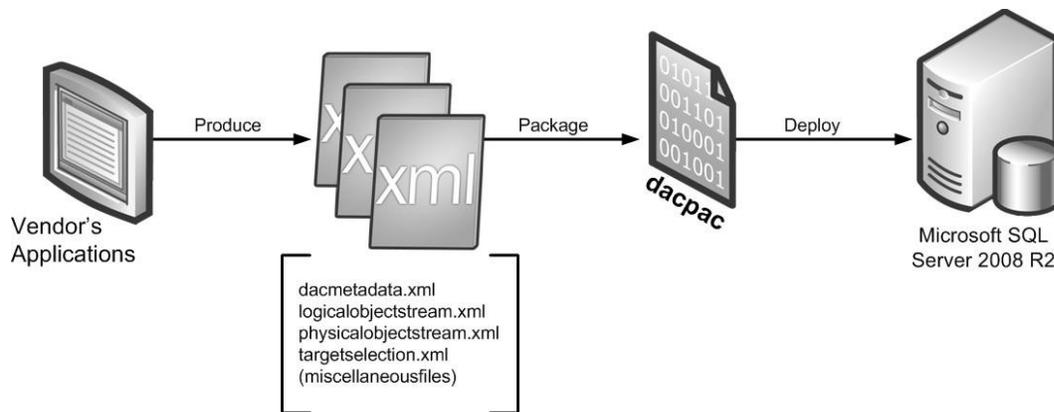
Figure 3: Import data

A vendor can package the XML parts to a .dacpac file by using the API that is specified in System.IO.Packaging [MSDN-PACKNAME] and can deploy the .dacpac file to SQL Server 2008 R2 by using the DAC API. To create a .dacpac file that can be deployed to SQL Server 2008 R2, a vendor's .dacpac file must contain dacmetadata.xml, logicalobjectstream.xml, physicalobjectstream.xml, and, optionally, targetselection.xml.

2.2.1 Data Description

Customer data

The customer data is a schema of a vendor's proprietary data to be imported into a Microsoft® SQL Server® 2008 R2 database. In this version, supported objects must be specified in [MSDN-DACSUPOB].

Intended user

The intended user is a vendor who can import a vendor's proprietary data to a SQL Server 2008 R2 database by using the .dacpac file format.

2.2.2   Format and Protocol Summary

The following table provides a comprehensive list of the formats and protocols used in an import data portability scenario.

Protocol or format name	Description	Reference
Data-Tier Application File (.dacpac) Format	The data-tier application file format is a package of XML files that serves as the packaging format for the data-tier application.	[MS-DACPAC]
Microsoft.SqlServer.Management.DAC Namespace	The **Microsoft.SqlServer.Management.Dac** namespace contains classes that represent the DAC objects.	[MSDN-DACAPI]
System.IO.Packaging Namespace	The **System.IO.Packaging** namespace provides classes that support storage of multiple data objects in a single container.	[MSDN-PACKNAME]

2.2.3   Data Portability Methodology

The data portability methodology describes the packaging and deployment steps to take when using DAC API. A vendor must provide its proprietary methodology to produce XML parts to be packaged in a .dacpac file. The XML parts and .dacpac files that are produced by the vendor's proprietary methodology must be compatible with [MS-DACPAC].

Package a data-tier application

To package a data-tier application, follow these steps:

1. Initialize a new instance of the **System.IO.Packaging.Package** class [MSDN-PACKNAME].

2. Create a **PackagePart** class for the XML part file stream in the package [MSDN-PACKPARTCON]. **PackageParts** must include logicalobjectstream.xml, physicalobjectstream.xml, dacmetadata.xml, and, optionally, targetselection.xml, as specified in [MS-DACPAC].

3. Close the package. The package must be saved with the *.dacpac file name extension [MSDN-PACKNAME].

Deploy a data-tier application

To deploy a data-tier application, load the .dacpac file, and then install it to a Microsoft® SQL Server® 2008 R2 database [MSDN-DACTYPE]. For more information, refer to the **DacStore.Install** method [MSDN-DACSTIN].

2.2.3.1   Preconditions

A SQL Server user must be a member of the **dbcreator** fixed server role and have ALTER ANY LOGIN server permission on the Microsoft® SQL Server® 2008 R2 instance to deploy the .dacpac file.

A vendor must create .dacpac file XML parts that are compatible with the format that is specified in [MS-DACPAC].

A .dacpac file created by a vendor must be compatible with the package format that is specified in [MSDN-PACKNAME].

2.2.3.2 Versioning

This version of import data scenario is applicable to Microsoft SQL Server 2008 R2.

2.2.3.3 Error Handling

Data-tier application error handling and troubleshooting are described in [MSDN-DACERROR].

2.2.3.4 Coherency Requirements

Imported data must be specified in SQL Server object list [MSDN-DACSUPOB].

2.2.3.5 Additional Considerations

There are no additional considerations.

3 Change Tracking Page

No table of changes is available. The document is either new or has had no changes since its last release.

4   Index

C

G

R

T

Release: Wednesday, February 9, 2011