[MS-DPBACPAC]:
Data-Tier Application Schema and Data Data Portability Overview

Revision Summary

Date	Revision History	Revision Class	Comments
07/07/2011	0.1	New	Released new document.
11/03/2011	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/19/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
02/23/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
03/27/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
05/24/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
06/29/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.

Release: Monday, July 16, 2012

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-2]. A .dacpac file consists of a package of XML parts that represent metadata of the data-tier application and SQL Server object **schema** [MS-DACPAC]. The BACPAC file format extends the DACPAC file format to include table data, in addition to schema data.

This document provides an overview of data portability scenarios that describe exporting and importing data between Microsoft® SQL Server® 2012 and a vendor's application by using a .bacpac file as a portable artifact. In these scenarios, a vendor must provide an API or XML transformation methodology to produce or consume the .bacpac file within the vendor's application, unless the .bacpac file is implemented by using the Microsoft DAC API [MSDN-DACAPI-2]. Additionally, the vendor needs to provide an API or other means to de-serialize encoded schema or table data unless the API is implemented by using the Microsoft DAC API.
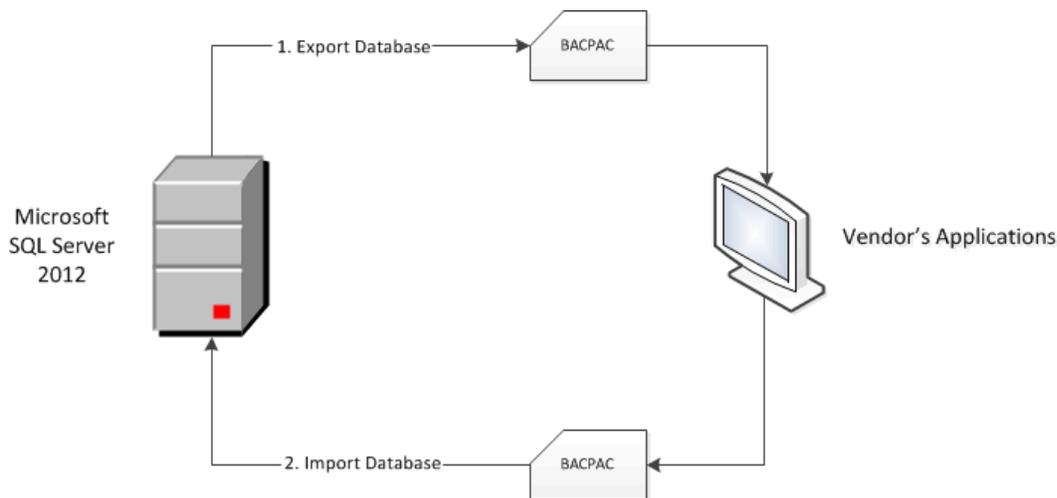
Figure 1: Conceptual overview of export and import data portability

In the Export Database scenario in the preceding figure, a vendor can implement an application by using the DAC API, as specified in [MSDN-DACAPI-2], to export a SQL Server database to a .bacpac file. The methodology for exporting a SQL Server database to a .bacpac file is described in section 2.1.

In the Import Database scenario in the preceding figure, a vendor can implement an application by using the DAC API to import the vendor-produced .bacpac file into SQL Server 2012. This methodology is described in section 2.2.

1.1 Glossary

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

database
schema

The following terms are specific to this document:

JavaScript Object Notation (JSON): A text-based, data interchange format that is used to transmit structured data, typically in Asynchronous JavaScript + XML (AJAX) Web applications, as described in [RFC4627]. The JSON format is based on the structure of ECMAScript (Jscript, JavaScript) objects.

1.2 References

[MS-BACPAC] Microsoft Corporation, "Data-Tier Application Schema and Data File (.bacpac) Format Structure Specification".

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

[MSDN-DACAPI-2] Microsoft Corporation, "Microsoft.SqlServer.Management.DAC Namespace", http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.management.dac%28v=SQL.110%29.aspx

[MSDN-DACSTOR] Microsoft Corporation "DacStore Class", http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.management.dac.dacstore(SQL.110).aspx

[MSDN-DacStore.Export] Microsoft Corporation, "BACPAC Export Interface Documentation", http://msdn.microsoft.com/en-us/library/dd146365(SQL.100).aspx

[MSDN-DacStore.Import] Microsoft Corporation, "DacStore.Import" http://msdn.microsoft.com/en-us/library/dd146365(SQL.100).aspx

[MSDN-DACSUPOB-2] Microsoft Corporation, "SQL Server Objects Supported in Data-tier Applications", http://msdn.microsoft.com/en-us/library/ee210549(SQL.110).aspx

[MSDN-DBSTATE-2] Microsoft Corporation, "Database States", http://msdn.microsoft.com/en-us/library/ms190442%28v=SQL.110%29.aspx

[MSDN-JSONSer] Microsoft Corporation, "JSON Serialization", http://msdn.microsoft.com/en-us/library/bb410770.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-TDAC] Microsoft Corporation, "Troubleshooting Data-tier Applications", http://msdn.microsoft.com/en-us/library/ee240741(SQL.110).aspx

[MSDN-UNDERDAC-2] Microsoft Corporation, "Understanding Data-tier Applications", http://msdn.microsoft.com/en-us/library/ee240739(SQL.110).aspx

[RFC4627] Crockford, D., "The application/json Media Type for Javascript Object Notation (JSON)", RFC 4627, July 2006, http://www.ietf.org/rfc/rfc4627.txt

2   Data Portability Scenarios

2.1   Export Data

The data export scenario describes exporting a customer database from Microsoft® SQL Server® 2012 to a .bacpac file so that a vendor can consume it within the vendor's application. As shown in the following figure, a .bacpac file can be created by exporting a Microsoft® SQL Server® database and then unzipping the .bacpac file. A vendor can consume the XML parts of the contents of the .bacpac file as a native XML format. In this case, the vendor must implement the methodology to consume the .bacpac within the vendor's application.

As shown in the following figure, the .bacpac file consists of dacmetadata.xml, logicalobjectstream.xml, physicalobjectstream.xml, bacpacmetadata.xml, and **JavaScript Object Notation (JSON)**-encoded table data. Additionally, it may contain targetselection.xml and miscellaneous files, such as Transact-SQL scripts. For more information about the file format structure, see [MS-BACPAC].
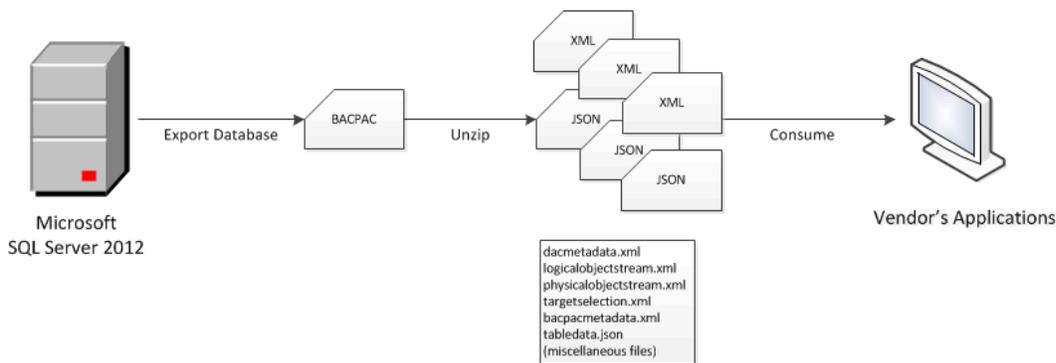
Figure 2: Export data

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

2.1.1   Data Description

Customer data

The customer data is a schema representation of a Microsoft® SQL Server® database and instances in Microsoft® SQL Server® 2012. In this version, the .dacpac file supports a subset of SQL Server objects, as specified in [MSDN-DACSUPOB-2].

Intended user

The intended user is a vendor who can export SQL Server object schema from SQL Server 2012 to a .dacpac 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.

Release: Monday, July 16, 2012

Format name	Description	Short name
Data-Tier Application File (BACPAC) Format	The BACPAC file format is a package of XML files and JSON-encoded table data that serves as the packaging format for an exported database.	[MS-BACPAC]
Microsoft.SqlServer.Management.DAC Namespace	The **Microsoft.SqlServer.Management.Dac** namespace contains classes that represent the DAC objects.	[MSDN-DACAPI-2]
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 by using the DAC API and **System.IO.Packaging**. The vendor's proprietary implementation for consuming the .dacpac is outside the scope of this section.

Export to a .bacpac file

To export a Microsoft® SQL Server® database to a .bacpac file, follow these steps:

1. Initialize a new instance of the **DacStore** class with an open connection to the target SQL Server instance. For more information, see [MSDN-DACSTOR].

2. Export the database to the specified file path. For more information, see the **DacStore.Export** method [MSDN-DacStore.Export].

Unzip a .bacpac file

To unzip a .bacpac file by using **System.IO.Packaging**, follow these steps:

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

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

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

2.1.3.1   Preconditions

The Microsoft® SQL Server® database must be ONLINE as specified in [MSDN-DBSTATE-2].

2.1.3.2   Versioning

This version of the export database scenario is applicable to Microsoft® SQL Server® 2012.

2.1.3.3   Error Handling

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

2.1.3.4 Coherency Requirements

The Microsoft® SQL Server® object must be listed as a supported object in [MSDN-DACSUPOB-2].

2.1.3.5 Additional Considerations

There are no additional considerations.

2.2 Import Data

The data import scenario describes importing vendor data to a .bacpac file so that the data can be deployed to Microsoft® SQL Server® 2012 as a data-tier application. As shown in the following figure, a vendor can produce XML parts that conform to the BACPAC [MS-BACPAC] structure format and package it to a .bacpac file. Note that the vendor must implement the methodology that produces the XML parts and the JSON-encoded table data within the vendor's application.

Produce → JSON XML → Package → BACPAC → Import Database → Microsoft SQL Server 2012

Vendor's Applications

dacmetadata.xml
logicalobjectstream.xml
physicalobjectstream.xml
targetselection.xml
bacpacmetadata.xml
tabledata.json
(miscellaneous files)
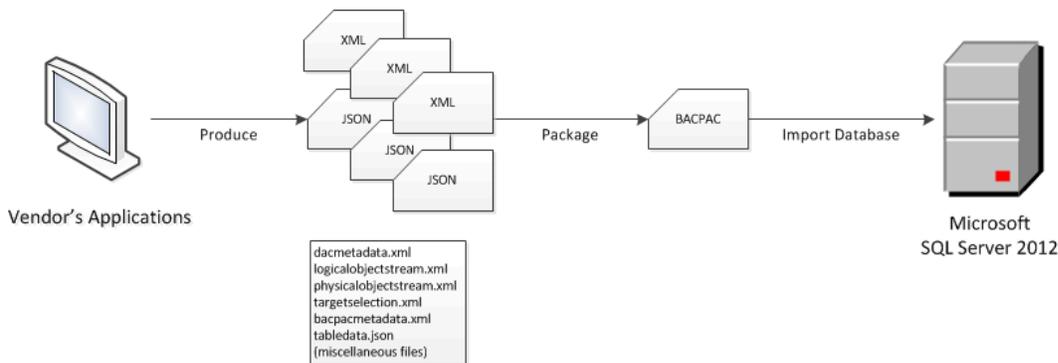
Figure 3: Import data

A vendor can package the XML parts to a .bacpac file by using the API that is specified in **System.IO.Packaging** [MSDN-PACKNAME] and can import the .bacpac file to SQL Server 2012 by using the DAC API. To create a .bacpac file that can be deployed to SQL Server 2012, the vendor's .bacpac file must contain dacmetadata.xml, logicalobjectstream.xml, physicalobjectstream.xml, bacpacmetadata.xml, JSON-encoded table data, 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® 2012 database. In this version, supported objects must be specified in [MSDN-DACSUPOB-2].

For supported data types, customer data is serialized into the .bacpac by using JSON [MSDN-JSONSer].

Intended user

The intended user is a vendor who can import a vendor's proprietary data to a SQL Server 2012 database by using the BACPAC 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.

Format name	Description	Short name
Data-Tier Application File (BACPAC) Format	The BACPAC file format is a package of XML files and JSON-encoded table data that serves as the packaging format for an exported database.	[MS-BACPAC]
Microsoft.SqlServer.Management.DAC Namespace	The **Microsoft.SqlServer.Management.Dac** namespace contains classes that represent the DAC objects.	[MSDN-DACAPI-2]
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 the DAC API. A vendor must provide its proprietary methodology to produce XML and table data to be packaged in a .bacpac file. The XML parts, table data, and .bacpac file that are produced by the vendor's proprietary methodology must be compatible with the BACPAC file format [MS-BACPAC].

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 file stream in the package [MSDN-PACKPARTCON]. **PackagePart** classes must include logicalobjectstream.xml, physicalobjectstream.xml, dacmetadata.xml, bacpacmetadata.xml, tabledata.json, and, optionally, targetselection.xml, as specified in [MS-BACPAC].

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

Deploy a data-tier application

To deploy a data-tier application, load the .bacpac file, and then import it to a Microsoft® SQL Server® 2012 database. For more information about the **DacStore.Import** method, see [MSDN-DacStore.Import].

2.2.3.1 Preconditions

A Microsoft® 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® 2012 instance to import the .bacpac.

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

A .bacpac 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 the import data scenario is applicable to Microsoft® SQL Server® 2012.

2.2.3.3 Error Handling

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

2.2.3.4 Coherency Requirements

Imported data must be specified in the Microsoft® SQL Server® object list [MSDN-DACSUPOB-2].

2.2.3.5 Additional Considerations

There are no additional considerations.

3 Change Tracking

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

4 Index

C

D

G

R

T