PSQL v13 ADO.NET Data Provider Release Notes
General Release – June 2017

Contents

This file contains the following topics:

About PSQL ADO.NET Data Provider

PSQL v13 provides two versions of the ADO.NET Data Provider: 4.2 and 4.3. All versions are installed by default with the database engine.

PSQL ADO.NET Data Provider 4.2 and 4.3 add support for the combinations with Microsoft .NET Framework and Microsoft Entity Framework as shown in the following table. Each row of the table represents the compatible combinations of versions of these three products.

PSQL Provider	Microsoft .NET Framework	Microsoft Entity Framework
ADO.NET Data Provider 4.2	2.0, 3.0, 3.5, 3.5 SP1, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2	—
ADO.NET Data Provider 4.3	2.0, 3.0, 3.5, 3.5 SP1, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2	—
ADO.NET Entity Framework Provider 4.2	4.5, 4.5.1, 4.5.2, 4.6.1, 4.6.2	5.0
ADO.NET Entity Framework Provider 4.3	4.5, 4.5.1, 4.5.2, 4.6.1, 4.6.2	5.0
ADO.NET Entity Framework Provider 4.2.0.6	4.5, 4.5.1, 4.5.2, 4.6.1, 4.6.2	6.0, 6.0.1, 6.0.2
ADO.NET Entity Framework Provider 4.3.0.6	4.5, 4.5.1, 4.5.2, 4.6.1, 4.6.2	6.0, 6.0.1, 6.0.2, 6.1, 6.1.1, 6.1.2

The PSQL ADO.NET 4.2 and 4.3 SDKs are integrated with Microsoft Visual Studio 2012, 2013, and 2015.

The PSQL ADO.NET Entity Framework data provider supports three versions of the Microsoft Entity Framework: 5.0, 6.0, and 6.1.

All versions listed apply to both 32- and 64-bit versions of .NET Framework.

If you are using ADO.NET without customization, then code written for earlier versions of the .NET Framework and of the PSQL data provider is compatible with PSQL data provider 4.2 and 4.3.

To use PSQL ADO.NET Entity Framework Provider 4.2 or 4.3, your applications must target .NET Framework 4.5 or later.

Microsoft Entity Framework 6.1, 6.1.1, are 6.1.2 are compatible only with PSQL ADO.NET Entity Framework Provider 4.3.0.6.

What's Deprecated in This Release?

- The PSQL ADO.NET data provider release 4.2 does not provide integration for Microsoft Visual Studio 2008 or 2010

Getting Started

The files required for the PSQL Data Provider for ADO.NET are provided through two different installations. The Provider assemblies are included by default when the PSQL database engine (all editions) is installed. The design-time files, sample source code, templates, and libraries, are included in the software development kit (SDK) installation for the ADO.NET access method.

Provider Assemblies

The installation of the database engine copies the Provider assemblies to version-specific directories under the PSQL\bin directory in PSQL installation location. Separate directories are used for 4.2 and 4.3.

If the installation detects a supported version of the .NET Framework, the following also occurs:

- The Provider assemblies are installed to the Windows Global Assembly Cache (GAC) as assembly versions 4.2.0.0 and 4.3.0.0.
- The .NET Framework DbProviderFactories namespace is updated with the following Provider instances:
  - Pervasive.Data.SqlClient.4.2
  - Pervasive.Data.SqlClient.4.3
  - Pervasive.Data.SqlClient

Note The instances are added to the DbProviderFactories name space only for final, GA, releases of the .NET Framework. The name space is *not* updated if the .NET Framework installed on the system is any release prior to the GA, such as alpha, beta or RC.

The generic provider instance (Pervasive.Data.SqlClient) always points to the latest provider assembly.

The provider assemblies copied to the PSQL\bin directory are not precompiled at installation time. That is, they are not added to the Windows native image service on the local computer. If you include the provider assemblies as private, side-by-side assemblies installed with your application and want them added to the Windows native image service on the target system, your installation must add them. See the topic "Native Image Service" in the Microsoft MSDN Library.

Design-Time Files

The design-time files needed for the different PSQL Membership Providers for ASP.NET and Visual Studio integration are available for download from the Actian website. Each assembly version has its own SDK installation executable.

The membership providers should be installed on the computer where .NET application development will be performed, and also on the client computers that will execute the application.

Each membership provider is installed to the GAC during the installation process.

Referencing the Different Providers

In addition to installing the sample source code files, templates, and libraries, the SDK installation executable also performs the following integration for the PSQL ADO.NET Data Provider:

- The 4.2 Provider integrates with Visual Studio 2012, 2013, and 2015.
- Integrates the Provider with Windows Performance Monitor (perfmon).
- Integrates the Provider with Visual Studio Server Explorer.
- All of the SDK installation executables for the PSQL ADO.NET Data Provider copy Item and Project templates to the appropriate Visual Studio templates folders when integrating with Visual Studio. The 4.2 Provider also includes Model First templates in addition to the Item and Project templates.

Installing Visual Studio After Installing the PSQL ADO.NET Data Provider

If you install Visual Studio after installing the PSQL ADO.NET SDK, rerun the SDK installation executable. The SDK installation integrates the PSQL Membership Provider with Visual Studio.

Logging Application Block Configuration

For the Logging Application Block to function correctly, you must include the following line in your application:

```
ConfigurationManager.GetSection("Pervasive.Data.SqlClient.Entity");
```

Design Considerations

The following information can help you decide how you want to design your application to function with the PSQL ADO.NET Data Provider.

Design Method	Comments
Application references generic provider instance of PSQL Provider	• Application can use updates to the provider released by Actian without being recompiled. • Application does not need to distribute the provider files. • Application always uses the latest assembly version installed in the GAC by the PSQL installation and updated by PSQL patches.
Application references a version-specific provider instance such as PSQL Provider PSQL Provider 4.2.	• Application always uses the latest patches for a particular version of the provider installed in the GAC. • Application will not use new releases of the provider installed by PSQL.

Application references the provider's strong name (a name that consists of an assembly's text name, version number, and culture information (if provided), with a public key and a digital signature generated over the assembly).	• Application distributes that specific version of the provider. • Application installs the specific provider as a private side-by-side assembly in the application installation folder.

Membership Provider for ASP.NET 2.0

The ASP.NET Login controls use the custom PSQL Membership Provider to specify a separate database for storing user information.

Additional information about writing a custom membership provider is available at http://msdn2.microsoft.com/en-us/library/f1kyba5e.aspx.

See Libraries for the location of the Membership Provider.

The sample ASP.NET application included with the libraries and samples shows how to create users, change passwords, recover passwords, and validate users.

Libraries

The SDK installation executable copies the following libraries to version-specific directories under the PSQL\bin directory. (See also "Where are the PSQL files installed?" in *Getting Started with PSQL*.)

Library	Description
PSQL.Membership.Provider.dll	Allows the ASP.NET Login controls to use a PSQL database.
PSQL.VisualStudio.DataTools.dll	Used by the VS.NET development environment for Server Explorer integration.

Membership Provider Requirements

- PSQL v13 database engine
- PSQL.Membership.Provider.dll
- Database using V2 metadata and a table of user information. The table also must be defined with V2 metadata.

► Steps to use the Membership Provider

- Create the PSQL database using V2 metadata. (See "To create a new database" in *PSQL User's Guide*.)
- Create the table using the "Users Table Script" below.
- Modify the membership section settings in WEB.CONFIG. See "Example Modified WEB.CONFIG" below.

Login Controls within ASP.NET will now use the PSQL Membership Provider.

Users Table Script

```
CREATE TABLE Users
```

```
(
PKID UniqueIdentifier NOT NULL PRIMARY KEY,
Username Char (255) NOT NULL,
ApplicationName char (255) NOT NULL,
Email Char (128) NOT NULL,
Comment Char (255),
"Password" Char (128) NOT NULL,
PasswordQuestion Char (255),
PasswordAnswer Char (255),
IsApproved SmallInt,
LastActivityDate DateTime,
LastLoginDate DateTime,
LastPasswordChangedDate DateTime,
CreationDate DateTime,
IsOnLine SmallInt,
IsLockedOut SmallInt,
LastLockedOutDate DateTime,
FailedPasswordAttemptCount Integer,
FailedPasswordAttemptWindowStart DateTime,
FailedPasswordAnswerAttemptCount Integer,
FailedPasswordAnswerAttemptWindowStart DateTime
)
```

Example Modified WEB.CONFIG

```
<connectionStrings>
<add name="PsqlServices" connectionString="ServerDSN=ASPNETPSQL;" /> </connectionStrings> <system.web>
<membership defaultProvider="PsqlMembershipProvider" userIsOnlineTimeWindow="120">
<providers>
<clear />
<add
name="PsqlMembershipProvider"
type="PSQL.Membership.PsqlMembershipProvider"
connectionStringName="PsqlServices"
enablePasswordRetrieval="true"
enablePasswordReset="true"
requiresQuestionAndAnswer="true"
writeExceptionsToEventLog="true"
debug="false"
passwordFormat="Clear"/>
</providers>
```

```
</membership>
</system.web>
```

IPv6 Environment

The PSQL ADO.NET data providers support IPv6 environments (IPv4/IPv6 mixed or IPv6-only).

Documentation

To help with the development of your .NET applications, see *PSQL Data Provider for .NET Guide* and *PSQL Programmer's Guide* in the PSQL SDK documentation.

Samples

The following table summarizes the samples provided with the ADO.NET Data Provider. By default, the SDK installation executable copies the sample files to version-specific directories under the Application Data\Actian\PSQL directory. Note that the application data location varies depending on the bit-architecture of the operating system. See "Where are the PSQL files installed?" in *Getting Started with PSQL*.

Folder	Sample	Purpose of Sample
Provider	CommandBuilder	Creating a command from an SQL statement
	CreateDatabaseObjects	Creating database objects such as a table or stored procedure
	DataAdapter	Using the data adapter to read or modify data
	DataGridUpdate	Updating data in a grid control
	DataReader	Using the data reader to get data
	Escapes	Processing escape functions in statements
	LimitingRows	Limiting the number of rows returned in a result set
	LocalTransactions	Processing a transaction
	LongDataTypes	Using long data types such as LONGVARCHAR
	ScalarValue	Executing scalar functions to return a value
	StoredProcedures	Using stored procedures

MembershipProvider	ASPX files for using the membership provider with the ASP.NET Login controls	Various login activities, changing password, and so forth
Enterprise Library	Sample using Microsoft Enterprise Library	Demonstrating the use of Microsoft Enterprise Library

Known Issues and Usage Notes

All known issues for PSQL v13 are published on the Actian website.

The following additional notes apply to using the PSQL v13 ADO.NET Data Provider:

- **Entity Framework Applications Upgrade from EF5 to EF6 – Code First**

  While working with Entity Framework applications, upgrading from Microsoft ADO.NET Entity Framework v5.0 (EF5) to v6.0 (EF6) against the same database may cause issues in the Code First workflows. Thereby you receive NotSupported exception. To resolve, drop the __MigrationHistory table and then rerun the application.

- **Code First Migration – Map to Stored Procedure**

  In the Code First Migration workflow, with the stored procedure mappings enabled, running the Enable-Migration command creates a <Timestamp>_InitialCreate.cs file in the Configuration folder, which causes build errors.

  To resolve this issue, delete <Timestamp>_InitialCreate.cs. Its deletion has no effect on data loss or migrations.

- **Code First Migration – On Tables Containing Data**

  Performing Code First migrations on a table that contains data in the back end returns a "File is locked" error. This is because in the Code First migration workflow of Entity Framework, all queries are executed inside a local transaction (generally Alter Commands). This behavior is not supported by PSQL.

- **Code First – Set Initalizers**

  When two or more Entity applications try to connect to same database, where at least one context contains set initializers other than "Create Database If Not Exist," data may be lost. To resolve this issue, you can use the customizable Migration History feature and give the new Migration History table a different name from the existing table in the back end.

- **Customizing the Migrations History Table – Renaming __MigrationHistory table**

  While renaming the __MigrationHistory table, be sure that the HistroyContext primary key length is within the 254-byte maximum length allowed by PSQL.

- **Customizing the Migrations History Table – Resizing Key Column Lengths**

  For Code First migrations, when you resize the default primary key columns of the customizable Migration History table, you must adhere to the maximum allowed primary key length set by the PSQL server. Otherwise, the provider sets the default length values as follows:

  If Unicode is enabled, then length of

  - MigrationId is 42.

- ContextKey is 83.

If Unicode is not enabled, then length of

- MigrationId is 85.
- ContextKey is 169.

- **Customizing the Migrations History Table – Renaming Migration History Table**

  For a Code First model, renaming the Migration History table may cause issues while working with DropCreateDatabaseIfModelChanges or DropCreateDatabaseAlways Set Initializers.

  Progress recommends that you rename the Migration History table at another time.

- **Customizing the Migrations History Table – Adding New Columns**

  Due to third-party limitations, you cannot modify the size of any column after adding new columns to the customizable Migration History table.

- **Code First Migration – Add/Drop Multiple Columns or Alter Column to Contexts with Relationships**

  Adding multiple columns or dropping multiple columns or altering single or multiple columns in a single migration, where the context has relationships, results in Btrieve Error 88. This is a known issue for PSQL.

- **Lightswitch 2011 and 2012 Support**

  Lightswitch loads the DateTime control for the PSQL Date column. When you load a table with a Date type column to be used in a Lightswitch application, Lightswitch loads the DateTime control for it. Since storing of time values is not supported, the time entered for the field is not preserved and the application resets the time portion to 00:00. As a workaround, change the Lightswitch mapping for this column in the Lightswitch designer to Date instead of DateTime.

- **Entity Framework Support**

  The Entity Framework provider does not support mapping for UniqueIdentifier type. When you choose a table that contains a column of type UniqueIdentifier while generating an entity data model, it returns the error "UniqueIdentifier data type is not supported." The table is still imported in the entity model but, without the column of type UniqueIdentifier. You can continue to work with this table in an Entity Framework application if the column of type UniqueIdentifier is nullable.

- **Installing on a Machine with Earlier Versions of the Data Providers**

  The PSQL data provider for ADO.NET 4.2 and 4.3 can exist on the same machine with earlier released versions of the data provider (GA version or with any of the patches or service packs). However, it cannot be installed in the same directory.

- **PSQL ADO.NET Data Provider Missing from Perfmon List**

  If you do not find the installed ADO.NET data provider listed in the Performance Object list of Perfmon, do the following:

  - Ensure that the data provider was installed with Administrator rights.
  - Run the .NET application that references the ADO.NET data provider.

  The data provider should then be available in the drop-down list.

- **Logging Application Block Configuration**

  In order for the Logging Application Block to function correctly, you must include the following line in your application:

  ```
  ConfigurationManager.GetSection("Pervasive.Data.SqlClient.Entity");
  ```

- **Installing Visual Studio After Installing the PSQL ADO.NET Data Provider**

To install Visual Studio for the first time on a machine that has the PSQL ADO.NET data provider installed, you must use the Visual Studio Integration Tool.

- To install the Visual Studio Integration Tool

  At a command prompt, run the InstallProviderIntegration.cmd script, located in the Tools folder. Specify the installation directory of the Pervasive.Data.SqlClient.dll.

- To uninstall the Visual Studio Integration Tool

  At a command prompt, run the UninstallProviderIntegration.cmd script, located in the Tools folder.

Technical Support

You can obtain technical support from several online options at the Actian website:

- Knowledge Base. Search hundreds of articles for answers and solutions others have found useful.
- Community Forums. Join a technical discussion or post a question to start a new one.
- PSQL Database Support. Open a service ticket, submit a defect, or purchase support.

Disclaimer

ACTIAN CORPORATION LICENSES THE SOFTWARE AND DOCUMENTATION PRODUCT TO YOU OR YOUR COMPANY SOLELY ON AN "AS IS" BASIS AND SOLELY IN ACCORDANCE WITH THE TERMS AND CONDITIONS OF THE ACCOMPANYING LICENSE AGREEMENT.