IBM DB2 10.5
for Linux, UNIX, and Windows

Developing ADO.NET and OLE DB Applications

Updated October, 2014

IBM

SC27-4549-01

IBM DB2 10.5
for Linux, UNIX, and Windows

Developing ADO.NET and OLE DB Applications

Updated October, 2014

SC27-4549-01

Note

Before using this information and the product it supports, read the general information under Appendix B, "Notices," on page 87.

Contents

Chapter 1. ADO.NET application development

In recent years, Microsoft has been promoting a new software development platform for Windows, known as the .NET Framework. The .NET Framework is Microsoft's replacement for Component Object Model (COM) technology. The following points highlight the key .NET Framework features:

- You can code .NET applications in over forty different programming languages. The most popular languages for .NET development are C# and Visual Basic .NET.
- The .NET Framework class library provides the building blocks with which you build .NET applications. This class library is language agnostic and provides interfaces to operating system and application services.
- Your .NET application (regardless of language) compiles into Intermediate Language (IL), a type of bytecode.
- The Common Language Runtime (CLR) is the heart of the .NET Framework, compiling the IL code on the fly, and then running it. In running the compiled IL code, the CLR activates objects, verifies their security clearance, allocates their memory, executes them, and cleans up their memory once execution is finished.

Through these features, the .NET Framework facilitates a wide variety of application implementations (for example, Windows forms, web forms, and web services), rapid application development, and secure application deployment. COM and COM+ proved to be inadequate or cumbersome for all the aforementioned features.

The .NET Framework provides extensive data access support through ADO.NET. ADO.NET supports both connected and disconnected access. The key component of disconnected data access in ADO.NET is the DataSet class, instances of which act as a database cache that resides in your application's memory.

For both connected and disconnected access, your applications use databases through what's known as a data provider. Various database products include their own .NET data providers for, including DB2® for Windows.

A .NET data provider features implementations of the following basic classes:

- Connection: Establishes and manages a database connection.
- Command: Executes an SQL statement against a database.
- DataReader: Reads and returns result set data from a database.
- DataAdapter: Links a DataSet instance to a database. Through a DataAdapter instance, the DataSet can read and write database table data.

Microsoft provides two data providers, the OLE DB .NET Data Provider and ODBC .NET Data Provider. The OLE DB .NET Data Provider is a bridge provider that feeds ADO.NET requests to the IBM® OLE DB Provider (by way of the COM interop module). ODBC .NET Data Provider is a bridge provider that feeds ADO.NET requests to the IBM ODBC Driver. These .NET data provider are not recommended for access to DB2 family databases. The IBM Data Server Provider for .NET is a high performance, managed ADO.NET data provider. This is the recommended .NET data provider for use with DB2 family databases. ADO.NET database access using the IBM Data Server Provider for .NET has fewer restrictions, and provides significantly better performance than the OLE DB and ODBC .NET bridge providers.

Deploying .NET applications (Windows)

To simplify .NET application deployment, IBM provides the IBM Data Server Driver Package, a small-footprint client that is ideal for use in mass deployment scenarios.

You can use the IBM Data Server Runtime Client instead, if the additional features of that client are required over the IBM Data Server Driver Package.

Before you begin

- Before deployment, you must build your .NET application, which you can do with either Visual Studio or the command line. For more information about building .NET applications, see the related tasks.
- Computers that you use to build .NET applications and computers where you will deploy .NET applications must have a supported version of the Windows operating system, in addition to other software, as described in "Supported .NET development software":
  - Build systems
    - Windows operating system
    - Visual Studio
    - .NET Framework Redistributable Package
    - .NET Framework Software Development Kit
  - Deployment systems
    - Windows operating system
    - .NET Framework Redistributable Package

Procedure

To deploy a .NET application:

1. Install the IBM Data Server Driver Package onto the computers where you will deploy your application. During the installation, set the IBM Data Server Driver Package installation to be the default database client interface copy.

   Note: Any existing database applications that run against an IBM data server will use this new installation of the IBM Data Server Driver Package. Test those applications against the new driver before rolling out your deployed .NET application.
2. Install your built application onto the computers where your application will run.

Supported .NET development software

IBM Data Server Provider for .NET supports the following .NET Framework and Visual Studio versions.

Supported development software for .NET Framework applications

In addition to an IBM data server client or driver package, you need one of the supported tools to develop .NET Framework applications:

- Visual Studio 2008
- Visual Studio 2010

- Visual Studio 2012

Supported deployment software for .NET Framework applications

In addition to an IBM data server client or driver package, you need one of the following packages to deploy .NET Framework applications. In most cases, a .NET Framework redistributable package is included with a Windows installation.

- .NET Framework Version 2.0 Redistributable Package
- .NET Framework Version 3.0 Redistributable Package
- .NET Framework Version 3.5 Redistributable Package
- .NET Framework Version 4.0 Redistributable Package
- .NET Framework Version 4.5 Redistributable Package

When you install a 64-bit IBM Data Server Package, both 32-bit and 64-bit providers are installed and configured.

If a .NET Framework is not installed, the IBM Data Server Client and driver installer will not install the IBM Data Server Provider for .NET. You must install the IBM Data Server Provider for .NET manually.

DB2 integration in Visual Studio

The IBM Database Add-Ins for Visual Studio component is a collection of IBM database development features that integrate seamlessly into your Visual Studio development environment.

The IBM Database Add-Ins for Visual Studio component presents a simple interface to IBM databases. For example, you can create database objects with the designer and wizard tools instead of SQL statements. You can use the integrated DB2 SQL editor to write SQL statements. The DB2 SQL editor contains the following features:

- Colored SQL text for increased readability
- Integration with the Microsoft Visual Studio IntelliSense feature, which provides for intelligent auto-completion while you are typing DB2 scripts

You can perform following tasks with the IBM Database Add-Ins for Visual Studio component:

- Open various DB2 development and administration tools.
- Create and manage DB2 projects in the Solution Explorer.
- Access and manage DB2 data connections from the Server Explorer.
- Create and modify DB2 scripts, including scripts to create stored procedures, functions, tables, views, indexes, and triggers.

Visual Studio 2008, 2010 and 2012

The IBM Database Add-Ins for Visual Studio component is included as a separately installable component that can be installed after IBM data server client product is installed. Your environment must already have the Microsoft Visual Studio software installed or installation of the IBM Database Add-Ins for Visual Studio component cannot be completed successfully.

Attention: The IBM Database Add-Ins for Visual Studio component is not supported by all editions of Microsoft Visual Studio software. You must ensure that the edition of Microsoft Visual Studio software you are using supports the use of external Add-Ins.

You can download the IBM Database Add-Ins for Visual Studio product for different DB2 versions and fix packs from the Download DB2 Fix Packs by version for DB2 for Linux, UNIX, and Windows websiteDownload DB2 for Linux, UNIX, and Windows Fix Packs by version for DB2 for Linux, UNIX, and Windows website (www.ibm.com/support/docview.wss?rs=71&uid=swg27007053).

IBM Data Server Provider for .NET

The IBM Data Server Provider for .NET extends database server support for the ADO.NET interface. The provider delivers high-performing, secure access to IBM data servers.

The IBM Data Server Provider for .NET is a name that is used to describe the .NET providers that are packaged with the IBM data server clients products. There are two .NET providers included in the IBM data server clients or IBM Data Server Driver Package. The two .NET providers are also called the Common .NET Providers.

The DB2 .NET provider

You can use the DB2 .NET provider to access all supported DB2 database servers and Informix® database servers. To connect to DB2 for z/OS® and IBM DB2 for IBM i servers, you require the DB2 Connect™ Server license.

The dynamic-link library file for the DB2 .NET provider is `IBM.Data.DB2.dll`.

For information about supported DB2 and Informix database servers, see the detailed system requirements for a specific product site (http://pic.dhe.ibm.com/infocenter/prodguid/v1r0/clarity/softwareReqsForProduct.html).

The Informix .NET provider

Important: The Informix .NET provider (IBM.Data.Informix.dll) is deprecated since DB2 Version 10.1 Fix Pack 1 and might be discontinued in a later release. Start using the DB2 .NET provider (IBM.Data.DB2.dll) to connect to Informix database servers.

You can use the Informix .NET provider to access supported Informix servers.
The dynamic-link library file for the Informix .NET provider is `IBM.Data.Informix.dll`.
For information about supported Informix database servers, see the detailed system requirements for a specific product site (http://pic.dhe.ibm.com/infocenter/prodguid/v1r0/clarity/softwareReqsForProduct.html).

To develop and run applications that use the IBM Data Server Provider for .NET, you need the .NET Framework.

You can also use the IBM Database Add-Ins for Visual Studio software to help quickly and easily develop .NET applications for IBM data servers with Microsoft Visual Studio. You can use the IBM Database Add-Ins for Visual Studio software to create database objects such as indexes and tables and develop server-side objects such as stored procedures and user-defined functions.

testconn command

The **testconn** command can be used to test the database connectivity of the DB2 .NET data provider.

There are two versions of the **testconn** command. The **testconn20** command is used to test the DB2 .NET data provider for .NET Framework 2.0 and the **testconn40** command is used to test the DB2 .NET data provider for .NET Framework 4.0.

In 64-bit IBM data server product environments, both 32-bit and 64-bit versions of the **testconn** command are available:

- The **testconn20** command is the 64-bit version of a **testconn** command for the .NET framework 2.0. In 32-bit IBM Data Server product environments, there is only one **testconn20** command, which is 32-bit.
- The **testconn40** command is the 64-bit version of a **testconn** command for the .NET framework 4.0. In 32-bit IBM Data Server product environments, there is only one **testconn40** command, which is 32-bit.
- The **testconn20_32** command is the 32-bit version of a **testconn** command for the .NET framework 2.0. The **testconn20_32** command is only available in 64-bit IBM Data Server product environments.
- The **testconn40_32** command is the 32-bit version of a **testconn** command for the .NET framework 4.0. The **testconn40_32** command is only available in 64-bit IBM Data Server product environments.

Command syntax

```
>>--+-testconn20-+--+--+-------------------------------------+--conn-string--+-------><
    '-testconn40-'  |  +--dtc--------------------------------+               |
                    |  +--ids--------------------------------+               |
                    |  +--idsold-----------------------------+               |
                    |  +--+--validateExtendedInsightWithConnect-+-+          |
                    |  |  '--validateEIWithConnect-------------' |          |
                    |  +--maxStep--step-number-----------------+               |
                    |  +--specific----------------------------+               |
                    |  '--+--enumerateremotedbs-+-------------'               |
                    |     '--enumremotedbs------'                             |
                    +--+--validateExtendedInsight-+---------------------------+
                    |  '--validateEI--------------'                           |
                    +--asp-----------------------------------------------------+
                    +--discover------------------------------------------------+
                    +--+--enumeratedbs-+----------------------------------------+
                    |  '--enumdbs------'                                      |
                    '--+--enumeratelocaldbs-+-----------------------------------'
                       '--enumlocaldbs------'
```

Command parameters

-dtc

Tests the connection for distributed transaction.

-ids

Tests the connection with the Common Informix .NET provider (IBM.Data.Informix.dll).

Important: The Informix .NET provider (IBM.Data.Informix.dll) is deprecated since DB2 Version 10.1 Fix Pack 1 and might be discontinued in a later release. Start using the DB2 .NET provider (IBM.Data.DB2.dll) to connect to Informix database servers.

`-idsold`

Tests the connection with Informix client SDK .NET provider. The `-idsold` option requires the Informix client SDK.

`-validateExtendedInsight` or `-validateEI`

Validates the `db2dsdriver.cfg` file structure and OPM EI monitoring configuration for the database that is mentioned in the connection string.

`-validateExtendedInsightwithConnect` or `-validateEIwithConnect`

Validates the `db2dsdriver.cfg` file structure and OPM EI monitoring configuration for the database that is mentioned in the connection string. Also, a connection is established to the database and reports the status of database monitoring.

`-maxStep` *step-number*

Runs only the specified number of validation steps in the **`testconn`** command. The **`testconn`** command runs validation tests in six steps:

- Step 1: Prints the IBM .NET data provider version and the .NET framework version information.
- Step 2: Validates the `db2dsdriver.cfg` file.
- Step 3: Tests the connection to the specified database in the connection string.
- Step 4: Validates the presence of packages by selecting rows from the SYSIBM.SYSTABLES table.
- Step 5: Validates the presence of schema functions by calling the GetSchema() method.
- Step 6: Tests the XA connection to the specified database in the connection string. This step is only run when the `-dtc` option is specified.

`-specific`

Tests the connection with the fix pack specific provider that is present under the `netfXX/specific` directory.

`-enumerateremotedbs` or `-enumremotedbs`

Lists the available databases on the remote server that is specified in the connection string.

`-asp`

Tests the database connection for an ASP .NET application.

`-discover`

Lists the DB2 servers that are accessible through the DB2 administration server (DAS). The `-discover` option requires following IBM Data Server product installation:

- The IBM database server product.
- The IBM Data Server Client software.
- The IBM Data Server Runtime Client software.

-enumeratedbs or **-enumdbs**

Enumerates the databases that are accessible from the computer where the command is issued. The `-enumeratedbs` or `-enumdbs` option requires following IBM Data Server product installation:

- The IBM database server product.
- The IBM Data Server Client software.
- The IBM Data Server Runtime Client software.

-enumeratelocaldbs or **-enumlocaldbs**

Lists the databases available on the local computer. The `-enumeratelocaldbs` or `-enumlocaldbs` option requires following IBM Data Server product installation:

- The IBM database server product.
- The IBM Data Server Client software.
- The IBM Data Server Runtime Client software.

conn-string

Specifies a connection string, which contains all information that is needed to connect to a target database.

There are total of six validation steps that are associated with **testconn** command. The sixth validation step is only called if `-dtc` option is specified. Following **testconn** command output example lists all six validation steps:

```
C:\Program Files\IBM\IBM DATA SERVER DRIVER\bin>testconn20 -dtc
"database=sampledsn;uid=username;pwd=password"
adding MSDTC step

Step 1: Printing version info
        .NET Framework version: X.X.XXXXX.XXXX
        64-bit
        DB2 .NET provider version: X.X.X.X
        DB2 .NET file version: X.X.X.X
        Capability bits: ALLDEFINED
        Build: XXXXXXXX
        Factory for invariant name IBM.Data.DB2 verified
        Factory for invariant name IBM.Data.Informix verified
        IDS.NET from DbFactory is Common IDS.NET
        VSAI is not installed properly
        Elapsed: 1.2969165

Step 2: Validating db2dsdriver.cfg against db2dsdriver.xsd schema file
        C:\ProgramData\IBM\DB2\IBMDBCL1\cfg\db2dsdriver.cfg against
C:\ProgramData\IBM\DB2\IBMDBCL1\cfg\db2dsdriver.xsd
        Elapsed: 0

Step 3: Connecting using "database=sampledsn;uid=username;pwd=password"
        Server type and version: DB2/NT XX.XX.XXXX
        Elapsed: 2.8594665

Step 4: Selecting rows from SYSIBM.SYSTABLES to validate existence of packages
   SELECT * FROM SYSIBM.SYSTABLES FETCH FIRST 5 rows only
        Elapsed: 0.3281355

Step 5: Calling GetSchema for tables to validate existence of schema functions
        Elapsed: 0.906279

Step 6: Creating XA connection
        DB2TransactionScope: Connection Closed.
        Elapsed: 3.2657295


Test passed.
```

You can also run only the first four validation steps by specifying the `-maxStep` *step-number* option:

```
C:\Program Files\IBM\IBM DATA SERVER DRIVER\bin>testconn20 -maxStep 4 -dtc
"database=sampledsn;uid=username;pwd=password"
max step 4
adding MSDTC step

Step 1: Printing version info
        .NET Framework version: X.X.XXXXX.XXXX
        64-bit
        DB2 .NET provider version: X.X.X.X
        DB2 .NET file version: X.X.X.X
        Capability bits: ALLDEFINED
        Build: XXXXXXXX
        Factory for invariant name IBM.Data.DB2 verified
        Factory for invariant name IBM.Data.Informix verified
        IDS.NET from DbFactory is Common IDS.NET
        VSAI is not installed properly
        Elapsed: 1.2969165

Step 2: Validating db2dsdriver.cfg against db2dsdriver.xsd schema file
        C:\ProgramData\IBM\DB2\IBMDBCL1\cfg\db2dsdriver.cfg against
C:\ProgramData\IBM\DB2\IBMDBCL1\cfg\db2dsdriver.xsd
        Elapsed: 0

Step 3: Connecting using "database=sampledsn;uid=username;pwd=password"
        Server type and version: DB2/NT XX.XX.XXXX
        Elapsed: 2.8594665

Step 4: Selecting rows from SYSIBM.SYSTABLES to validate existence of packages
   SELECT * FROM SYSIBM.SYSTABLES FETCH FIRST 5 rows only
        Elapsed: 0.1875024


Test passed.
```

Programming applications to use the IBM Data Server Provider for .NET

Programming applications to use the IBM Data Server Provider for .NET requires understanding of available features in the IBM Data Provider for .NET. After you determine the requirement of your application, you can leverage available features in the IBM Data Provider for .NET.

Generic coding with the ADO.NET common base classes

The .NET Framework, versions 2.0, 3.0, and 3.5, features a namespace called `System.Data.Common`, which features a set of base classes that can be shared by any .NET data provider. This facilitates a generic ADO.NET database application development approach, featuring a constant programming interface across different databases.

The following C# demonstrates a generic approach to establishing a database connection.

```
DbProviderFactory factory = DbProviderFactories.GetFactory("IBM.Data.DB2");
DbConnection conn = factory.CreateConnection();
DbConnectionStringBuilder sb = factory.CreateConnectionStringBuilder();

if( sb.ContainsKey( "Database" ) )
{
   sb.Remove( "database" );
   sb.Add( "database", "SAMPLE" );
}

conn.ConnectionString = sb.ConnectionString;

conn.Open();
```

The `DbProviderFactory` object is the point where any generic ADO.NET application begins. This object creates generic instances of .NET data provider objects, such as connections, data adapters, commands, and data readers, which work with a specific database product. In the previous example, the "IBM.Data.DB2" string passed into the GetFactory method uniquely identifies the IBM Data Server Provider for .NET, and results in the initialization of a DbProviderFactory instance that creates database provider object instances specific to the IBM Data Server Provider for .NET.

The DbConnection object can connect to DB2 family Informix databases, just as a DB2Connection object, which is actually inherited from DbConnection. Using the DbConnectionStringBuilder class, you can determine the connection string keywords for a data provider, and generate a custom connection string. The code in the previous example checks if a keyword named "database" exists in the IBM Data Server Provider for .NET, and if so, generates a connection string to connect to the SAMPLE database.

Connecting to a database from an application using the IBM Data Server Provider for .NET

When using the IBM Data Server Provider for .NET, a database connection is established through the DB2Connection class.

Procedure

To connect to a database:

1. Create a string that stores the connection parameters. The format for a typical connection string format is:

   ```
   Server=<ip address/localhost>:<port number>;
   Database=<db name>;
   UID=<userID>;
   PWD=<password>;
   Connect Timeout=<Timeout value>
   ```

   Examples of possible connection strings are:

   Example 1:

   ```
   String connectString = "Database=SAMPLE";
   // When used, attempts to connect to the SAMPLE database.
   ```

   Note: If you specify only the database name in the connection string, the other information such as the server, userid, and password, must be included in the db2dsdriver.cfg file.

   Example 2:

   ```
   String cs = "Server=srv:50000;
   Database=SAMPLE;
   UID=db2adm;
   PWD=ab1d;Connect Timeout=30";
   // When used, attempts to connect to the SAMPLE database on the server
   // 'srv' through port 50000 using 'db2adm' and 'ab1d' as the user id and
   // password respectively. If the connection attempt takes
   // more than thirty seconds, the attempt will be terminated and an error
   // will be generated.
   ```

2. Pass the connectString to the DB2Connection constructor.
   - Connecting to a database in C#:

     ```
     String connectString = "Database=SAMPLE";
     DB2Connection conn = new DB2Connection(connectString);
     conn.Open();
     return conn;
     ```

   - Connecting to a database in Visual Basic .NET:

     ```
     Dim connectString As String = "Database=SAMPLE"
     Dim conn As DB2Connection = new DB2Connection(connectString)
     conn.Open()
     Return conn
     ```

3. Use the DB2Connection object's Open method to formally connect to the database identified in connectString.

Connection pooling with the IBM Data Server Provider for .NET

When a connection is first opened against a DB2 database, a connection pool is created. As connections are closed, they enter the pool, ready to be reused within the same process by other applications that need connections.

The IBM Data Server Provider for .NET uses a normalized set of connection string attributes for determining the connection pool. By using normalized attributes, the chances of an application reusing connections is increased.

The IBM Data Server Provider for .NET enables connection pooling by default.

Note: You can turn connection pooling off using the `Pooling=false` connection string keyword/value pair. However, if you turn off connection pooling COM+ applications will not work.

You can control the behavior of the connection pool by setting following connection string keywords:

- The minimum and maximum pool size (**`MinPoolSize`** and **`MaxPoolSize`**)
- The length of time a connection can be idle before it is returned to the pool (**`ConnectionLifetimeInPool`**)

Creating a trusted connection with IBM Data Server Provider for .NET

You can create a trusted connection with the .NET provider with the **`TrustedContextSystemUserID`** and **`TrustedContextSystemPassword`** connection string keywords.

The following keywords are available in the connection string:

- **`TrustedContextSystemUserID`**, or **`tcsuid`**, which specifies the trusted context SYSTEM AUTHID to be used with the connection.
- **`TrustedContextSystemPassword`**, or **`tcspwd`**, which specifies the password corresponding to the trusted context SYSYTEM AUTHID to be used with the connection.

If the **`TrustedContextSystemPassword`** keyword is specified without a **`TrustedContextSystemUserID`** keyword value, an InvalidArgument exception is thrown. The **`UserID`** keyword is also required in a trusted context scenario.

IBM Data Server Provider for .NET supports trusted context with DB2 for Linux, UNIX, and Windows and DB2 for z/OS servers.

Example

Suppose a trusted context has been established on the server with the following information:

```
CREATE TRUSTED CONTEXT ctxName1
BASED UPON CONNECTION USING SYSTEM AUTHID masteruser
ATTRIBUTES ( PROTOCOL 'TCPIP',
             ADDRESS '9.26.146.201',
             ENCRYPTION 'NONE' )
ENABLE
WITH USE FOR userapp1 WITH AUTHENTICATION, userapp2 WITH AUTHENTICATION;
```

The SYSTEM AUTHID, masteruser, has a corresponding password, masterpassword. Each specific user/application, userapp1, and userapp2, has a corresponding password, passapp1 and passapp2.

In order to use this trusted context, applications would issue connection strings as follows:

- Application 1

  ```
  database=db;server=server1:446;
  UserID=userapp1;Password=passapp1;
  TrustedContextSystemUserID=masteruser;TrustedContextSystemPassword=masterpassword
  ```

- Application 2

  ```
  database=db;server=server1:446;
  UserID=userapp2;Password=passapp2;
  TrustedContextSystemUserID=masteruser;TrustedContextSystemPassword=masterpassword
  ```

Note: The UserID keyword corresponds to the end user of the connection in a trusted context situation, just as in standard applications.

Following .NET program open and close a connection:

```
[C#]
DB2Connection conn = new DB2Connection();

conn.ConnectionString = "database=db;server=server1:446;"
     + "UserID=userapp1;Password=passapp1;"
     + "TrustedContextSystemUserID=masteruser;"
     + "TrustedContextSystemPassword=masterpassword;"

conn.Open();

// Do processing as userapp1, such as querying tables

conn.Close();

conn.ConnectionString = "database=db;server=server1:446;UserID=userapp2;"
     + "Password=passapp2;TrustedContextSystemUserID=masteruser;"
     + "TrustedContextSystemPassword=masterpassword;"

conn.Open();

// Do processing as userapp2

conn.Close();
```

If the trusted context processing fails because no trusted context was set up on the server, or the server does not support trusted contexts, an error with SQLCODE CLI0197E will be thrown. If the TrustedContextSystemUserID keyword value is invalid (too long, for example), an error with SQLCODE CLI0124E will be thrown. The server might report an error with SQLCODE SQL1046N, SQL30082N, or SQL0969N with a native error code of -20361. Any of these errors will cause Open() to fail.

Note: The trusted context processing happens on the next communication with the server.

SQL data type representation in ADO.NET database applications

ADO.NET applications can reference DB2 SQL data type values as parameter values and use these parameters values as part of SQL statement. You must reference appropriate IBM Data Server Provider for .NET data type values and .NET Framework data type values to prevent possible data truncation or data loss.

For specifying parameter values to be used as part of a SQL statement to be executed, IBM Data Server Provider for .NET objects must be used. The `DB2Parameter` object is used to represent a parameter to be added to a `DB2Command` object which represents a SQL statement. When specifying the data type value for the parameter, the IBM Data Server Provider for .NET data type values available in the `IBM.Data.DB2Types` namespace must be used. The `IBM.Data.DB2Types` namespace provides classes and structures to represent each of the supported DB2 SQL data types.

For local variables that might temporarily hold SQL data type values, appropriate IBM Data Server Provider for .NET data types, as defined in the `IBM.Data.DB2Types` Namespace, must be used.

The following table shows mappings between DB2Type data types, DB2 data types, Informix data types, Microsoft .NET Framework types, and DB2Types classes and structures.

Table 1. Mapping of DB2Types classes to DB2Type data types, DB2 data types, Informix data types, and .NET data types

Category	DB2Types Classes	DB2Type Data Type	DB2 Data Type	Informix Data Type	.NET Data Type
Binary data	DB2Binary	Binary	CHAR FOR BIT DATA		Byte[]
	DB2Binary	Binary[3]	BINARY		Byte[]
	DB2Binary	VarBinary[3]	VARBINARY		Byte[]
	DB2Binary	LongVarBinary[1]	LONG VARCHAR FOR BIT DATA		Byte[]
Character data	DB2String	Char	CHAR	CHAR	String
	DB2String	VarChar	VARCHAR	VARCHAR	String
	DB2String	LongVarChar[1]	LONG VARCHAR	LVARCHAR	String
Graphic data	DB2String	Graphic	GRAPHIC		String
	DB2String	VarGraphic	VARGRAPHIC		String
	DB2String	LongVarGraphic[1]	LONG VARGRAPHIC		String
LOB data	DB2Clob	Clob	CLOB	CLOB, TEXT	String
	DB2Blob	Blob	BLOB	BLOB, BYTE	Byte[]
	DB2Clob	DbClob	DBCLOB		String

1. These data types are not supported as parameters in DB2 .NET common language runtime routines.
2. A DB2ParameterClass.ParameterName property of the type DB2Type.Xml can accept variables of the following types: String, byte[], DB2Xml, and XmlReader.
3. These data types are applicable only to DB2 for z/OS and DB2 for i V6R1 and later.
4. This data type is only supported for DB2 for z/OS Version 9 and later releases and for DB2 for Linux, UNIX, and Windows Version 9.5 and later releases.
5. Date and Time objects can be timestamp string literals. Timestamp objects can be date string literals

Table 1. Mapping of DB2Types classes to DB2Type data types, DB2 data types, Informix data types, and .NET data types (continued)

Category	DB2Types Classes	DB2Type Data Type	DB2 Data Type	Informix Data Type	.NET Data Type
Numeric data	DB2Int16	SmallInt	SMALLINT	BOOLEAN, SMALLINT	Int16
	DB2Int32	Integer	INT	INTEGER, INT, SERIAL	Int32
	DB2Int64	BigInt, BigSerial	BIGINT	BIGINT, BIGSERIAL, INT8, SERIAL8	Int64
	DB2Real, DB2Real370	Real	REAL	REAL, SMALLFLOAT	Single
	DB2Double	Double	DOUBLE PRECISION	DECIMAL (≤ 29), DOUBLE PRECISION	Double
	DB2Double	Float	FLOAT	DECIMAL (32), FLOAT	Double
	DB2Decimal	Decimal	DECIMAL	MONEY	Decimal
	DB2DecimalFloat	DecimalFloat	DECFLOAT (16|34)[14]		Decimal
	DB2Decimal	Numeric	DECIMAL	DECIMAL (≤ 29), NUMERIC	Decimal
Date/Time data	DB2Date	Date	DATE	DATETIME (date precision)	DateTime String[5]
	DB2Time	Time	TIME	DATETIME (time precision)	TimeSpan String[5]
	DB2TimeStamp	Timestamp	TIMESTAMP	DATETIME (time and date precision)	DateTime String[5]
	DB2TimeStamp Offset	TimestampWith TimeZone	TIMESTAMP WITH TIME ZONE	N/A	DateTimeOffset String[5]
Row ID data	DB2RowId	RowId	ROWID		Byte[]
XML data	DB2Xml	Xml[2]	XML		Byte[]

Issuing SQL statements from a .NET application

You can issue SQL statements through a `DB2Command` class with its methods `ExecuteReader()` and `ExecuteNonQuery()`, and its properties `CommandText`, `CommandType`, and `Transaction`.

About this task

For SQL statements that produce output, you can use the `ExecuteReader()` method, and retrieve the results from a `DB2DataReader` object. For all other SQL statements, you can use the `ExecuteNonQuery()` method. You can initialize the `Transaction` property of the `DB2Command` object to a `DB2Transaction` object. A `DB2Transaction` object is responsible for rolling back and committing database transactions.

Issuing an UPDATE statement in C#:

```
// assume a DB2Connection conn
DB2Command cmd = conn.CreateCommand();
DB2Transaction trans = conn.BeginTransaction();
cmd.Transaction = trans;
cmd.CommandText = "UPDATE staff " +
                  "  SET salary = (SELECT MIN(salary) " +
                  "                  FROM staff " +
    "                 WHERE id &gt;= 310) " +
                  "  WHERE id = 310";

cmd.ExecuteNonQuery();
```

Issuing an UPDATE statement in Visual Basic .NET:

```
' assume a DB2Connection conn
DB2Command cmd = conn.CreateCommand();
DB2Transaction trans = conn.BeginTransaction();
cmd.Transaction = trans;
cmd.CommandText = "UPDATE staff " +
                  "  SET salary = (SELECT MIN(salary) " +
                  "                  FROM staff " +
    "                 WHERE id >= 310) " +
                  "  WHERE id = 310";
cmd.ExecuteNonQuery();
```

Issuing a SELECT statement in C#:

```
// assume a DB2Connection conn
DB2Command cmd = conn.CreateCommand();
DB2Transaction trans = conn.BeginTransaction();
cmd.Transaction = trans;
cmd.CommandText = "SELECT deptnumb, location " +
                  "  FROM org " +
                  "  WHERE deptnumb &lt 25";

DB2DataReader reader = cmd.ExecuteReader();
```

Issuing a SELECT statement in Visual Basic .NET:

```
' assume a DB2Connection conn
Dim cmd As DB2Command = conn.CreateCommand()
Dim trans As DB2Transaction = conn.BeginTransaction()
cmd.Transaction = trans
cmd.CommandText = "UPDATE staff " +
                  "  SET salary = (SELECT MIN(salary) " +
                  "                  FROM staff " +
    "                 WHERE id >= 310) " +
                  "  WHERE id = 310";

cmd.ExecuteNonQuery()
```

After your application performs a database transaction, you must either roll it back or commit it. The commit and rollback operation is done through the `Commit()` and `Rollback()` methods of a `DB2Transaction` object.

Rolling back or committing a transaction in C#:

```
// assume a DB2Transaction object conn
trans.Rollback();
...
trans.Commit();
```

Rolling back or committing a transaction in Visual Basic.NET:

```
' assume a DB2Transaction object conn
trans.Rollback()
...
trans.Commit()
```

The .NET data provider supports an application to retrieve the result sets from execution of anonymous blocks by using `DB2DataReader` or `DB2ResultSet` classes. For the .NET data provider to retrieve the result sets from anonymous block execution, the database server must support PL/SQL statements and the database must be enabled to process PL/SQL statements. The .NET data provider must declare cursors for the results sets that are returned from anonymous block execution by using the `BEGIN` statement, and not the`BEGIN COMPOUND` statement.

Retrieving a single result set from execution of anonymous block by using the `DB2DataReader` class in C#:

```
...
cmd.CommandText = "begin " +
                  "declare cursor1 cursor with return to client with hold for select c1 from t1; " +
                  "open cursor1; " +
                  "end;";
//Returns a result set by opened cursor cursor1
DB2DataReader dr = cmd.ExecuteReader();
while (dr.Read())
{
    //Process read data if required
}
dr.Close();
...
```

Retrieving a single result set from execution of anonymous block by using the `DB2ResultSet` class in C#:

```
...
cmd.CommandText = "begin " +
                  "declare cursor1 cursor with return to client with hold for select c1 from t1; " +
                  "open cursor1; " +
                  "end;";
//Returns a result set by opened cursor cursor1
DB2ResultSet ds = cmd.ExecuteResultSet(DB2CursorType.ForwardOnly);
while (ds.Read())
{
    //Process read data if required
}
ds.Close();
...
```

Retrieving multiple result sets from execution of anonymous block by using the `DB2DataReader` class in C#:

```
...
cmd.CommandText = " begin " +
                  "declare cursor1 cursor with return to client with hold for select c1 from t1; " +
                  "declare cursor2 cursor with return to client for select c2 from t2; " +
                  "open cursor1; " +
                  "open cursor2; " +
                  "end;";
//Returns multiple result sets by opened cursors
DB2DataReader dr = cmd.ExecuteReader();
while (dr.Read())
{
    //Process read data if required from cursor1
}
dr.NextResult(); //Get next result set
while (dr.Read())
{
    //Process read data if required from cursor2
}
dr.Close();
...
```

Retrieving multiple result sets from execution of anonymous block by using the DB2ResultSet class in C#:

```
...
cmd.CommandText = " begin " +
                "declare cursor1 cursor with return to client with hold for select c1 from t1; " +
                "declare cursor2 cursor with return to client for select c2 from t2; " +
                "open cursor1; " +
                "open cursor2; " +
                "end;";
//Returns multiple result sets by opened cursors
DB2ResultSet ds = cmd.ExecuteResultSet(DB2CursorType.ForwardOnly);
while (ds.Read())
{
    //Process read data if required from cursor1
}
ds.NextResult(); //Get next result set
while (ds.Read())
{
    //Process read data if required from cursor2
}
ds.Close();
...
```

Reading result sets from an application using the IBM Data Server Provider for .NET

When using the IBM Data Server Provider for .NET, the reading the result sets is done through a DB2DataReader object. The DB2DataReader method, Read() is used to advance to the next row in the result set.

About this task

The methods GetString(), GetInt32(), GetDecimal(), and other methods for all of the available data types are used to extract data from the individual columns of output. The DB2DataReader's Close() method is used to close the DB2DataReader object, which should always be done when reading the output is finished.

Reading a result set in C#:

```
// assume a DB2DataReader reader
Int16 deptnum = 0;
String location="";

// Output the results of the query
while(reader.Read())
{
  deptnum = reader.GetInt16(0);
  location = reader.GetString(1);
  Console.WriteLine("    " + deptnum + " " + location);
}
reader.Close();
```

Reading a result set in Visual Basic .NET:

```
' assume a DB2DataReader reader
Dim deptnum As Int16 = 0
Dim location As String ""

' Output the results of the query
Do While (reader.Read())
  deptnum = reader.GetInt16(0)
  location = reader.GetString(1)
  Console.WriteLine("    " & deptnum & " " & location)
Loop
reader.Close();
```

Calling stored procedures from .NET applications

.NET applications can call stored procedures with a DB2Command object.

Procedure

1. Make a connection to a target database. For steps to establish database connection, see "Connecting to a database from an application using the IBM Data Server Provider for .NET" on page 9.
2. Create the DB2Command object and set the CommandType property as either CommandType.StoredProcedure or CommandType.Text. The default value of the CommandType property is CommandType.Text. The CommandType.Text value can be used to call stored procedures. However, calling stored procedures is easier when you set the `CommandType` property to CommandType.StoredProcedure. When you use the CommandType.StoredProcedure object to call a stored procedure, you must specify the stored procedure name and parameters that are associated with the stored procedure. A stored procedure with same name and same parameters can exist under different schemas. To avoid calling an incorrect stored procedure, fully qualify the stored procedure name with the correct schema name. A C# code example of the CommandType.Text object follows:

   ```
   DB2Command cmd = conn.CreateCommand();
   String procCall = "CALL TEST_PROC (@input_param1)";
   cmd.CommandType = CommandType.Text;
   cmd.CommandText = procCall;
   ```

   Note: When the `CommandType` property is `CommandType.Text`, both `CALL` and `EXECUTE PROCEDURE` statements are supported.
   A C# code example of the `CommandType.StoredProcedure` object follows:

   ```
   DB2Command cmd = conn.CreateCommand();
   String procName = "TEST_PROC";
   cmd.CommandType = CommandType.StoredProcedure;
   cmd.CommandText = procName;
   ```

   Note: When the `CommandType` property is `CommandType.StoredProcedure`, named parameters are not supported.
3. Create the `DB2Command.Parameters` objects that correspond to the IN, INOUT and OUT parameters. If you are using parameter markers for stored procedure parameters, create the DB2Parameter objects and bind the DB2Parameter objects to the DB2Command.Parameters object with the Add method. A C# code example follows:

   ```
   DB2Parameter p1 = new DB2Parameter("input_param1", DB2Type.Integer);
   p1.Value = 123;
   db2Command.Parameters.Add(p1);
   ```

   You can pass the store procedure parameters with host variables, named parameters, or positioned parameters. However, you cannot use different methods to pass the stored procedure parameters within the same SQL statement. Parameters can be passed to the stored procedure in any order, when qualified by the parameter name as shown in following C# code example:

   ```
   CREATE PROCEDURE schema.my_proc ( IN var1 int, INOUT var2 int )
    LANGUAGE SQL
    BEGIN
     -- procedure code here
    END

   String procCall = "CALL my_proc (var2=>@param2, var1=>@param1");
   ```

   IBM Data Server Provider for .NET supports calling stored procedures with ARRAY data types as input (IN) parameters in following database servers:
   - DB2 for Linux, UNIX, and Windows.
   - DB2 for z/OS Version 11 server in new function mode (NFM).
   - DB2 for i V7R1 and later servers.

   ARRAY data types are not supported for the OUT and INOUT parameters. The ARRAY length value must be specified in the DB2Parameter.ArrayLength object for each ARRAY parameter. A C# code example follows:

```
Int32 integerArray = new Int32[] { 12, 34, 45, 67 };
DB2Parameter p1 = new DB2Parameter("input_param1", DB2Type.Integer);
p1.Value = integerArray;
p1.ArrayLength = 3;
db2Command.Parameters.Add(p1);
```

The Cursor enumeration member can be used when binding INOUT (InputOutput) or OUT (Output) parameters of the type cursor. A C# code example of output parameters follows:

```
DB2Command cmd = new DB2Command("cursor_test", conn)
cmd.CommandType = CommandType.StoredProcedure;
cmd.Parameters.Add("cursor1", DB2Type.Cursor).Direction =
ParameterDirection.Output;
cmd.Parameters.Add("cursor2", DB2Type.Cursor).Direction =
ParameterDirection.Output;

cmd.ExecuteNonQuery();
```

If your application is connecting to DB2 for z/OS Version 10 and later servers, your application must specify the correct data type for the input parameters of the stored procedure that you are calling. If your application specifies parameters that do not match the data type of the input parameter, an invalid conversion error is returned.

4. Run the DB2Command.ExecuteNonQuery() function to call a stored procedure. A C# code example follows:

```
cmd.ExecuteNonQuery();
```

If there are any OUT or INOUT parameters, you can obtain the parameter values with DB2DataReader object. A C# code example follows:

```
DB2DataReader  drOutput2 = cmd.Parameters[1].Value;
DB2DataReader  drOutput1 = cmd.Parameters[0].Value
```

IBM Data Server Provider for .NET saves extra network traffic that is associated with sending the implicit COMMIT statement when the following conditions are met:

- The connected database server is DB2 for z/OS Version 11 in new function mode (NFM).
- The BeginTransaction method is not called by the application.
- There are no open result-sets when the stored procedure completes the execution.

Example

A C# code with CommandType.Text example follows:

```
// assume a DB2Connection conn
DB2Transaction trans = conn.BeginTransaction();
DB2Command cmd = conn.CreateCommand();
String procName = "INOUT_PARAM";
String procCall = "CALL INOUT_PARAM (@param1, @param2, @param3)";
cmd.Transaction = trans;
cmd.CommandType = CommandType.Text;
cmd.CommandText = procCall;

// Register input-output and output parameters for the DB2Command
cmd.Parameters.Add( new DB2Parameter("@param1", "Value1");
cmd.Parameters.Add( new DB2Parameter("@param2", "Value2");
DB2Parameter param3 = new DB2Parameter("@param3", IfxType.Integer);
param3.Direction = ParameterDirection.Output;
cmd.Parameters.Add( param3 );

// Call the stored procedure
Console.WriteLine("  Call stored procedure named " + procName);
cmd.ExecuteNonQuery();
```

A Visual Basic code with CommandType.Text example follows:

```
' assume a DB2Connection conn
Dim trans As DB2Transaction = conn.BeginTransaction()
Dim cmd As DB2Command = conn.CreateCommand()
Dim procName As String = "INOUT_PARAM"
Dim procCall As String = "CALL INOUT_PARAM (?, ?, ?)"
cmd.Transaction = trans
cmd.CommandType = CommandType.Text
cmd.CommandText = procCall

' Register input-output and output parameters for the DB2Command
...

' Call the stored procedure
Console.WriteLine("  Call stored procedure named " & procName)
cmd.ExecuteNonQuery()
```

A C# code with CommandType.StoredProcedure example follows:

```
// assume a DB2Connection conn
DB2Transaction trans = conn.BeginTransaction();
DB2Command cmd = conn.CreateCommand();
String procName = "INOUT_PARAM";
cmd.Transaction = trans;
cmd.CommandType = CommandType.StoredProcedure;
cmd.CommandText = procName;

// Register input-output and output parameters for the DB2Command
...

// Call the stored procedure
Console.WriteLine("  Call stored procedure named " + procName);
cmd.ExecuteNonQuery();
```

A Visual Basic code with CommandType.StoredProcedure example follows:

```
' assume a DB2Connection conn
Dim trans As DB2Transaction = conn.BeginTransaction()
Dim cmd As DB2Command = conn.CreateCommand()
Dim procName As String = "INOUT_PARAM"
cmd.Transaction = trans
cmd.CommandType = CommandType.StoredProcedure
cmd.CommandText = procName

' Register input-output and output parameters for the DB2Command
...

' Call the stored procedure
Console.WriteLine("  Call stored procedure named " & procName)
cmd.ExecuteNonQuery()
```

A C# code example with ARRAY input parameter follows:

```
db2Command.CommandText = "arrayparamprocedure";
db2Command.CommandType = CommandType.StoredProcedure;
Int32 integerArray = new Int32[] { 12, 34, 45, 67 };
DB2Parameter p1 = new DB2Parameter("numbers_in", DB2Type.Integer);
p1.Value = integerArray;
p1.ArrayLength = 3;
String[] stringArray = new String[] {"i think i know", "but you never know", "how much i know" };
DB2Parameter p2 = new DB2Parameter("varchars_in", DB2Type.Varchar, 30);
p2.Value = stringArray;
p2.ArrayLength = 2;
db2Command.Parameters.Add(p1);
db2Command.Parameters.Add(p2);
db2Command.ExecuteNonQuery();
```

Simultaneously accessing the result sets returned by CURSOR type output parameters

When using the IBM Data Server Provider for .NET, the DB2Type.Cursor is specified to simultaneously access all the cursors in output parameters.

About this task

For Stored procedure that has multiple CURSOR type output parameters, binding DB2TYPE.Cursor to the output parameter object allows simultaneous access to all the cursors in output parameters.

For example, OrderDetails stored procedure declares three cursors, each giving relevant information about the product and its sales.

```
CREATE OR REPLACE TYPE cur AS CURSOR
CREATE PROCEDURE OrderDetails (p_startDate TIMESTAMP, p_endDate TIMESTAMP,
  OUT prodDetails cur, OUT prodOrderDetails cur, OUT custOrderDetails cur)
LANGUAGE SQL
BEGIN
  SET prodDetails = CURSOR WITH HOLD FOR
    SELECT p.pid, price, quantity FROM products p, inventory i
      WHERE p.pid = i.pid AND p.pid IN (SELECT DISTINCT pid FROM orders) ORDER BY pid;
  SET prodOrderDetails = CURSOR WITH HOLD FOR
    SELECT pid, COUNT(*), SUM (quantity) FROM orders
```

```
      WHERE date >= p_startDate AND date <= p_endDate GROUP BY pid ORDER BY pid;
  SET custOrderDetails = CURSOR WITH HOLD FOR
    SELECT pid, custID, COUNT(*), SUM(quantity) FROM orders
      WHERE date >= p_startDate AND date <= p_endDate
      GROUP BY pid, custID ORDER by pid, custID;
  OPEN prodDetails;
  OPEN prodOrderDetails;
  OPEN custOrderDetails;
END;
```

The caller needs to access the cursors simultaneously so that it can gather the relevant information for a particular product from each of the cursors and calculate the discount. To provide simultaneous access to the cursors, the stored procedure returns the cursors as output parameters. The application must set the DB2Type to DB2Type.Cursor when binding the CURSOR type output parameters for simultaneous access to occur.

```
//C# Code sample
cmd.CommandText = "CALL OrderDetails(
      @p_startDate, @p_endDate, @prodDetails, @prodOrderDetails, @custOrderDetails)";
cmd.Parameters.Add("@p_startDate", DateTime.Parse("1/1/2010"));
cmd.Parameters.Add("@p_endDate", DateTime.Parse("12/31/2010"));
cmd.Parameters.Add("@prodDetails", DB2Type.Cursor);
cmd.Parameters["@prodDetails"].Direction = ParameterDirection.Output;
cmd.Parameters.Add("@prodOrderDetails", DB2Type.Cursor);
cmd.Parameters["@prodOrderDetails"].Direction = ParameterDirection.Output;
cmd.Parameters.Add("@custOrderDetails", DB2Type.Cursor);
cmd.Parameters["@custOrderDetails"].Direction = ParameterDirection.Output;
cmd.ExecuteNonQuery();
DB2DataReader prodDetailsDR =
    (DB2DataReader)cmd.Parameters["@prodDetails"].Value;
DB2DataReader prodOrderDetailsDR =
    (DB2DataReader)cmd.Parameters["@prodOrderDetails"].Value;
DB2DataReader custOrderDetailsDR =
    (DB2DataReader)cmd.Parameters["@custOrderDetails"].Value;

while (prodOrderDetailsDR.Read())
{
    pid = prodOrderDetailsDR.GetInt32(0);
    numOrders = prodOrderDetailsDR.GetInt32(1);
    totalOrderQuantity = prodOrderDetailsDR.GetInt32(2);
    prodDetailsDR.Read();
    price = prodDetailsDR.GetDecimal(1);
    currentInventory = prodDetailsDR.GetInt32(2);
    int totalCustOrders = 0;
    while (custOrderDetailsDR.Read())
    {
        custID = custOrderDetailsDR.GetInt32(1);
        numOrdersByCust = custOrderDetailsDR.GetInt32(2);
        totalCustOrders += numOrdersByCust;
        totalOrderQuantityByCust = custOrderDetailsDR.GetInt32(3);
        //Calculate discount based on numOrders, numOrdersByCust,
        //  totalOrderQuantity, totalOrderQuantityByCust, price and currentInventory
        if (totalCustOrders == numOrders) //done with this pid
            break;
    }
}
prodDetailsDR.Close();
prodOrderDetailsDR .Close();
custOrderDetailsDR .Close();
```

The data reader from a cursor type output parameter can be accessed from the Value property only after invoking the ExecuteNonQuery method.

If the command is executed using either the ExecuteReader or ExecuteResultSet methods, the result sets are returned in the DB2DataReader or DB2ResultSet object. The subsequent result sets must be accessed sequentially by calling the NextResult method. Although the output parameters have been bound, accessing the output parameter Value property will result in an InvalidOperation exception because the query was not executed with the ExecuteNonQuery method.

When working with cursors simultaneously, the application might want to commit the work done before continuing with reading the cursor. For application to issue commit without destroying the open cursor, the cursor must be declared as holdable within the stored procedure.

Tracing IBM Data Server Provider for .NET

You can trace the activity of the IBM Data Server Provider for .NET by setting environment variables or by using an application configuration file.

About this task

You generate the .NET application trace by using the System.Diagnostics.Trace class. You can control the tracing of the .NET public methods and properties by using the System.Diagnostics.TraceSwitch class without recompiling or modifying the source code. Only the method and property calls that are explicitly made by the applications are traced. Internal method and property calls that are made by a .NET provider are not traced.

The **DB2NMPTRACE** trace switch property and the **DB2NMPTRACE** environment variable initializes the System.Diagnostics.TraceSwitch class for the DB2 .NET provider.

The **IFXNMPTRACE** trace switch property and the **IFXNMPTRACE** environment variable initializes the System.Diagnostics.TraceSwitch class for the Informix .NET provider.

Important: The Informix .NET provider (IBM.Data.Informix.dll) is deprecated since DB2 Version 10.1 Fix Pack 1 and might be discontinued in a later release. Start using the DB2 .NET provider (IBM.Data.DB2.dll) to connect to Informix database servers.

You can specify different levels of tracing by specifying one of the following System.Diagnostics.TraceLevel enumeration values for the System.Diagnostics.TraceSwitch class:

0	Turns off the tracing.
1	Turns on the tracing for errors.
2	Turns on the tracing for errors and warnings.
3	Turns on the tracing for errors, warnings, and informational messages.
4	Turns on the tracing for all messages.

Procedure

To trace IBM Data Server Provider for .NET:

1. Determine which .NET provider your application is using by checking the source code or contacting your application vendor. If your application references the IBM.Data.DB2 namespace, the application is using the DB2 .NET provider. If your application references the IBM.Data.Informix namespace, the application is using the Informix .NET provider.
2. Enable the .NET provider trace:
   - For applications that use the IBM.Data.DB2 namespace, use one of the following methods:
     - Set environment variables in the same session as the .NET application that you are tracing:
       a. Set the **DB2NMPTRACE** environment variable to a System.Diagnostics.TraceLevel enumeration value by using the **set** command. You can specify a value of `0 -4`, as explained in the "About this task" section. An example follows:
          ```
          set DB2NMPTRACE=1
          ```
       b. Specify the trace output directory by setting the **DB2NMPCONSOLE** environment variable with the **set** command. An example follows.
          ```
          set DB2NMPCONSOLE=c:\tmp\nmptrace
          ```
          The directory must exist, and you must have write permission for it.

- In the application configuration file, set the **DB2NMPTRACE** switch name element to a System.Diagnostics.TraceLevel enumeration value of `0 -4`. An example follows:

```
<system.diagnostics>
  <trace autoflush="false" indentsize="4">
    <listeners>
      <add name="configConsoleListener" type="System.Diagnostics.ConsoleTraceListener" />
    </listeners>
  </trace>
  <switches>
    <add name="DB2NMPTRACE" value="1" />
  </switches>
</system.diagnostics>
```

  The trace that you take by using the application configuration file is logged to a console.

- For applications that use the IBM.Data.Informix namespace, use one of the following two methods:
  - Set the environment variables in the same session as the .NET application that you are tracing:
    a. Set the **IFXNMPTRACE** environment variable to a System.Diagnostics.TraceLevel enumeration value by using the **set** command. You can specify a value of `0 -4`, as explained in the "About this task" section. An example follows:

    ```
    set IFXNMPTRACE=1
    ```

    b. Specify the trace output directory by specifying the **IFXNMPCONSOLE** environment variable with the **set** command. An example follows:

    ```
    set IFXNMPCONSOLE=c:\tmp\nmptrace
    ```

    The directory must exist, and you must have write permission for it.
  - In the application configuration file, set the **IFXNMPTRACE** switch name element to a System.Diagnostics.TraceLevel enumeration value of `0 - 4`. An example follows:

```
<system.diagnostics>
  <trace autoflush="false" indentsize="4">
    <listeners>
      <add name="configConsoleListener" type="System.Diagnostics.ConsoleTraceListener" />
    </listeners>
  </trace>
  <switches>
    <add name="IFXNMPTRACE" value="1" />
  </switches>
</system.diagnostics>
```

  The trace that you take by using the application configuration file is logged to a console.

Results

An example of trace output follows:

```
* * Started tracing program
* Creating connection
DB2Connection.DB2Connection1 api entry - database=nmpfvtu;
DB2Connection.DB2Connection1 api exit, rc = 0
* Opening connection
DB2Connection.Open api entry
DB2Connection.Open api exit, rc = 0
* Closing connection
DB2Connection.Close api entry
DB2Connection.Close api exit, rc = 0
* Ending program
DB2Connection.~DB2Connection api entry
DB2Connection.~DB2Connection api exit, rc = 0
DB2Connection.Dispose api entry
DB2Connection.Dispose api exit, rc = 0
```

The trace points that are dumped by the application are prefixed with an asterisk (*).

Optimizing queries in .NET applications using pureQuery

The .NET client drivers can leverage features found in pureQuery technology. These features enables existing .NET application queries to execute as static SQL. Static queries avoid the need to prepare certain statements at runtime. This can lead to improved security and performance in your applications.

For more information, see http://www.ibm.com/support/docview.wss?uid=swg27023946

The Microsoft Entity Framework support with the IBM Data Server Provider for .NET

You can generate EDM schemas, write and execute Entity SQL, and write and execute LINQ statements with the IBM Data Server Provider for .NET and the Microsoft Entity Framework.

The IBM Data Server Provider for .NET provides support for the Microsoft Entity Framework through the installation of the IBM Database Add-Ins for Visual Studio software. You must have an environment with a supported Microsoft Visual Studio software, an IBM data server client product and the IBM Database Add-Ins for Visual Studio installed.

Requirements for Microsoft Entity Framework support:

To use IBM Data Server Provider for .NET with Microsoft Entity Framework, you must have an environment with supported Microsoft Visual Studio software, an IBM data server client product, and the IBM Database Add-Ins for Visual Studio.

To use the Microsoft Entity Framework, you must have Microsoft .NET Framework 3.5 SP1 or later.

To manipulate entity data models by using the Microsoft Entity Data Model Wizard or Entity Designer, you also require Microsoft Visual Studio 2008 or later and the IBM Database Add-Ins for Visual Studio.

The Entity Framework 5.0 features require the Microsoft .NET Framework version that supports those features and the Microsoft Visual Studio 2012 or later software. Microsoft .NET Framework Version 4.0 or later is required for the following Entity Framework 5.0 features:

- Multiple diagrams per model
- Batch import of stored procedures

Microsoft .NET Framework Version 4.5 or later is required for the following Entity Framework 5.0 features:

- Enum support

Microsoft Entity Framework 5.0 support:

IBM Data Server Provider for .NET provides support for key Entity Framework 5.0 features.

IBM Data Server Provider for .NET supports the following Entity Framework 5.0 features with the IBM Database Add-Ins for Visual Studio and the Visual Studio 2012 or later software:

- **Enum support**: Enumeration of the Int16, Int32, and Int64 data types
- **Multiple diagrams per model**: Splitting of a model into multiple diagrams with the Entity Framework Designer (EF Designer)
- **Batch import of stored procedures**: Addition of multiple stored procedures during model creation

Supported canonical functions:

IBM Data Server Provider for .NET supports the canonical functions.

The following table lists the canonical functions that the IBM entity provider supports. Canonical functions are translated to the corresponding data source functions by the data provider.

Table 2. IBM entity provider support for canonical functions

Canonical function type	LINQ function	DB2 for Linux, UNIX, and Windows	DB2 for z/OS	DB2 for i	Informix
Aggregate	Average	Yes	Yes	Yes	Yes
	BigCount	Yes	Yes	Yes	Yes
	Count	Yes	Yes	Yes	Yes
	Maximum	Yes	Yes	Yes	Yes
	Minimum	Yes	Yes	Yes	Yes
	NewGuid	Yes*	Yes*	Yes*	Yes*
	StDev	Yes	Yes	Yes	Yes
	StDevP	Yes	Yes	Yes	Yes
	Sum	Yes	Yes	Yes	Yes
	Var	Yes	Yes	Yes	Yes
	VarP	Yes	Yes	Yes	Yes
Bitwise	BitWiseAnd	Yes	Yes*	Yes*	Yes
	BitWiseNot	Yes	Yes*	Yes*	Yes
	BitWiseOr	Yes	Yes*	Yes*	Yes
	BitWiseXor	Yes	Yes*	Yes*	Yes
Math	Abs	Yes	Yes	Yes	Yes
	Ceiling	Yes	Yes	Yes	Yes
	Floor	Yes	Yes	Yes	Yes
	Power	Yes	Yes	Yes	Yes
	Round (value,digits)	Yes	Yes	Yes	Yes
	Truncate (value,digits)	Yes	Yes	Yes	Yes
String	Concat	Yes	Yes	Yes	Yes
	Contains	Yes	Yes	Yes	Yes*
	EndsWith	Yes	Yes	Yes	Yes
	IndexOf	Yes	Yes	Yes	Yes*
	Left	Yes	Yes	Yes	Yes
	Length	Yes	Yes	Yes	Yes
	LTrim	Yes	Yes	Yes	Yes
	Replace	Yes	Yes	Yes	Yes
	Right	Yes	Yes	Yes	Yes
	RTrim	Yes	Yes	Yes	Yes
	StartsWith	Yes	Yes	Yes	Yes
	Substring	Yes	Yes	Yes	Yes
	ToLower	Yes	Yes	Yes	Yes
	ToUpper	Yes	Yes	Yes	Yes
	Trim	Yes	Yes	Yes	Yes

Table 2. IBM entity provider support for canonical functions (continued)

Canonical function type	LINQ function	DB2 for Linux, UNIX, and Windows	DB2 for z/OS	DB2 for i	Informix
Datetime	AddNanoseconds	Yes	Yes	Yes	Yes
	AddMicroseconds	Yes	Yes	Yes	Yes
	AddMilliseconds	Yes	Yes	Yes	Yes
	AddSeconds	Yes	Yes	Yes	Yes
	AddMinutes	Yes	Yes	Yes	Yes
	AddHours	Yes	Yes	Yes	Yes
	AddDays	Yes	Yes	Yes	Yes
	AddMonths	Yes	Yes	Yes	Yes
	AddYears	Yes	Yes	Yes	Yes
	CreateDateTime	Yes	Yes	Yes	Yes
	CreateDateTimeOffset		Yes		
	CurrentDateTimeOffset		Yes		
	CreateTime	Yes	Yes	Yes	Yes
	CurrentDateTime	Yes	Yes	Yes	Yes
	CurrentUtcDateTime	Yes	Yes	Yes	
	Day	Yes	Yes	Yes	Yes
	DayOfYear	Yes	Yes	Yes	Yes
	DiffNanoseconds	Yes	Yes	Yes	Yes*
	DiffMicroseconds	Yes	Yes	Yes	Yes*
	DiffMilliseconds	Yes	Yes	Yes	Yes*
	DiffSeconds	Yes	Yes	Yes	Yes*
	DiffMinutes	Yes	Yes	Yes	Yes*
	DiffHours	Yes	Yes	Yes	Yes*
	DiffDays	Yes	Yes	Yes	Yes*
	DiffMonths	Yes	Yes	Yes	Yes*
	DiffYears	Yes	Yes	Yes	Yes*
	GetTotalOffsetMinutes		Yes		
	Hour	Yes	Yes	Yes	Yes
	Millisecond	Yes	Yes	Yes	Yes
	Minute	Yes	Yes	Yes	Yes
	Month	Yes	Yes	Yes	Yes
	Second	Yes	Yes	Yes	Yes
	Truncate (datetime exp)	Yes	Yes	Yes	Yes
	Year	Yes	Yes	Yes	Yes

Important: Some of the canonical functions depend on the server. The SQL0440N* error indicates that your server does not support the specified function.

Limitations to Microsoft Entity Framework support:

There are known limitations to Microsoft Entity Framework support for the IBM Data Server Provider for .NET.

The following limitations apply:

General limitations:

- Only database-first scenarios are supported: any database object that you reference in Entity Framework must first exist in the database.
- Invocation of store-specific functions is not supported.

- Trusted context connection properties that you set in the Server Explorer Add Connection dialog are not passed to Entity Framework connections.

DB2 for z/OS server-specific limitations:

- Data type REAL is not supported. Applications must either use the FLOAT data type in the schema of the table or specify the type as FLOAT in the client schema (entity data model) even if the type on the server is REAL.

Informix Dynamic Servers

- Server-side pagination is not supported.

For information about Entity Framework limitations that are associated with updating the database model, see the troubleshooting technote (http://www-01.ibm.com/support/docview.wss?uid=swg21635456).

IBM Data Server Provider for .NET support for the Microsoft SQL Server Reporting Services

Microsoft SQL Server Reporting Services (SSRS) can connect to a DB2 database server by using an ODBC, OLE DB, or DB2 (IBM Data Server Provider for .NET) embedded connection type.

You can select DB2 (IBM Data Server Provider for .NET) as an embedded connection type in the SSRS application if one of the following DB2 products is installed in the SSRS environment:

- ADB2 database server product (for example, DB2 Enterprise Server Edition)
- IBM Data Server Client
- IBM Data Server Runtime Client
- IBM Data Server Driver Package

You can provide the credentials that are used in a connection to a DB2 database if you select the DB2 embedded connection in the supported Microsoft Visual Studio software. The credentials that you provide are masked instead of being displayed in plain text. Use of the DB2 embedded connection is supported for SSRS 2008 and SSRS 2012.

If you install the SSRS software in an environment where the DB2 product is already installed, by default, only ODBC and OLE DB embedded connection types are available for connections to DB2 databases. The reason is that the DB2 installation process updates the SSRS configuration files that are required to list IBM Data Server Provider for .NET in the embedded connection type section of the Microsoft Visual Studio software. If you installed the SSRS product after you installed the DB2 software, you cannot select the IBM Data Server Provider for .NET when you configure an embedded connection. If you want to select the IBM Data Server Provider for .NET for use, you must either reinstall the DB2 product or start the IBM Data Server configuration tool.

- To reinstall the DB2 product:
  1. Open the Control Panel.
  2. In the Programs and Features window, right-click the entry for the DB2 product.
  3. Click the **Repair** or **Change** button.
- To start the IBM Data Server Provider for .NET configuration tool:
  1. Open the **IBM DB2** group icon.
  2. Select the default DB2 copy. For example, `DB2COPY1`.

3. Select the **Set-up Tools** icon.
4. Select the **Configure DB2 .NET Data Provider** icon.

The IBM Data Server Provider for .NET configuration tool (**Configure DB2 .NET Data Provider**) is not available for the IBM Data Server Driver Package installation.

Using the Enterprise Library data access module

The Enterprise Library is a collection of application blocks designed to assist developers with common development challenges. Application blocks are provided as source code that can be used as is or modified for development projects.

The Enterprise Library data access module for IBM data servers can be obtained along with other modules at http://codeplex.com/entlibcontrib/SourceControl/PatchList.aspx.

For information about how to install and use the Enterprise Library data access module with IBM data servers (DB2, Informix database server, and U2), see the readme file found in the download package.

Resources

Below are several online resources that describe how to use the data access modules:

- EntLib Contrib Project Homepage: http://www.codeplex.com/entlibcontrib
- patterns & practices for Enterprise Library: http://www.codeplex.com/entlib
- Microsoft Enterprise Library Homepage: http://msdn.microsoft.com/en-us/library/cc467894.aspx
- IBM DB2 for .NET: http://www.ibm.com/software/data/db2/windows/dotnet.html

Building .NET Applications

Resources for building .NET applications.

Building Visual Basic .NET applications

DB2 products provide a bldapp.bat batch file for compiling and linking DB2 Visual Basic .NET applications.

This file is located in the sqllib\samples\.NET\vb directory along with sample programs that can be built with this file. The batch file takes one parameter, %1, for the name of the source file to be compiled (without the .vb extension).

About this task

This task will take you through the basic steps of building a Visual Basic .NET application using bldapp.bat with the DbAuth sample file.

Procedure

To build the program, `DbAuth`, from the source file, `DbAuth.vb`, enter:

```
bldapp DbAuth
```

To ensure you have the parameters you need when you run the executable, you can specify different combinations of parameters depending on the number entered:

1. No parameters. Enter just the program name:
   ```
   DbAuth
   ```
2. One parameter. Enter the program name plus the database alias:
   ```
   DbAuth <db_alias>
   ```
3. Two parameters. Enter the program name plus user ID and password:
   ```
   DbAuth <userid> <passwd>
   ```
4. Three parameters. Enter the program name plus the database alias, user ID, and password:
   ```
   DbAuth <db_alias> <userid> <passwd>
   ```
5. Four parameters. Enter the program name plus server name, port number, user ID, and password:
   ```
   DbAuth <server> <portnum> <userid> <passwd>
   ```
6. Five parameters. Enter the program name plus database alias, server name, port number, user ID, and password:
   ```
   DbAuth <db_alias> <server> <portnum> <userid> <passwd>
   ```

What to do next

To build and run the `LCTrans` sample program, you need to follow more detailed instructions given in the source file, `LCTrans.vb`.

Building C# .NET applications

DB2 products provide a bldapp.bat batch file for compiling and linking DB2 C# .NET applications. This batch file is located in the sqllib\samples\.NET\cs directory along with sample programs that can be built with this file.

The batch file takes one parameter, %1, for the name of the source file to be compiled (without the .cs extension).

About this task

This task will take you through the basic steps of building a C# .NET application using bldapp.bat with the DbAuth sample file.

Procedure

To build the program, `DbAuth`, from the source file, `DbAuth.cs`, enter:
```
bldapp DbAuth
```

To ensure you have the parameters you need when you run the executable, you can specify different combinations of parameters depending on the number entered:
1. No parameters. Enter just the program name:
   ```
   DbAuth
   ```
2. One parameter. Enter the program name plus the database alias:
   ```
   DbAuth <db_alias>
   ```
3. Two parameters. Enter the program name plus user ID and password:
   ```
   DbAuth <userid> <passwd>
   ```
4. Three parameters. Enter the program name plus the database alias, user ID, and password:
   ```
   DbAuth <db_alias> <userid> <passwd>
   ```

5. Four parameters. Enter the program name plus server name, port number, user ID, and password:

   ```
   DbAuth <server> <portnum> <userid> <passwd>
   ```
6. Five parameters. Enter the program name plus database alias, server name, port number, user ID, and password:

   ```
   DbAuth <db_alias> <server> <portnum> <userid> <passwd>
   ```

What to do next

To build and run the `LCTrans` sample program, you need to follow more detailed instructions given in the source file, `LCTrans.cs`.

Visual Basic .NET application compile and link options

This topic describes the various options available when compiling and linking Visual Basic .NET applications.

The following compile and link options are available for building Visual Basic .NET applications on Windows with the Microsoft Visual Basic .NET compiler, as demonstrated in the `bldapp.bat` batch file.

Note: The .NET Framework Version 1.1 is supported only with the .NET Provider Version 9.5 and earlier.

Compile and link options for stand-alone VB .NET applications using bldapp

Compile and link options for stand-alone VB .NET applications:

`%BLDCOMP%`
: Variable for the compiler. The default is `vbc`, the Microsoft Visual Basic .NET compiler.

`/r:"%DB2PATH%"\bin\%VERSION%\IBM.Data.DB2.dll`
: Reference the DB2 dynamic link library for the .NET framework version that you are using.

  `%DB2PATH%`
  : The `%DB2PATH%` variable represents root path of the DB2 product installation. The `%DB2PATH%` variable is not present on IBM Data Server Driver for ODBC and CLI or Data Server Driver Package installation. When using IBM IBM Data Server Driver for ODBC and CLI or Data Server Driver Package replace `%DB2PATH%` with a path where driver product is installed.

  `%VERSION%`
  : There are several supported versions of the .NET framework for applications. DB2 has a dynamic link library for each. For .NET Framework Version 2.0, 3.0, and 3.5, `%VERSION%` points to the `netf20\` sub-directory.

Compile and link options for the loosely-coupled sample program, `LCTrans` using bldapp:

`%BLDCOMP%`
: Variable for the compiler. The default is `vbc`, the Microsoft Visual Basic .NET compiler.

/out:RootCOM.dll
: Output the RootCOM dynamic link library, used by the LCTrans application, from the RootCOM.vb source file,

/out:SubCOM.dll
: Output the SubCOM dynamic link library, used by the LCTrans application, from the SubCOM.vb source file,

/target:library %1.cs
: Create the dynamic link library from the input source file (RootCOM.vb or SubCOM.vb).

/r:System.EnterpriseServices.dll
: Reference the Microsoft Windows System EnterpriseServices data link library.

/r:"%DB2PATH%"\bin\%VERSION%\IBM.Data.DB2.dll
: Reference the DB2 dynamic link library for the .NET framework version you are using.

  %DB2PATH%
  : The %DB2PATH% variable represents root path of the DB2 product installation. The %DB2PATH% variable is not present on IBM Data Server Driver for ODBC and CLI or Data Server Driver Package installation. When using IBM IBM Data Server Driver for ODBC and CLI or Data Server Driver Package replace %DB2PATH% with a path where driver product is installed.

  %VERSION%
  : There are several supported versions of the .NET framework for applications. DB2 has a dynamic link library for each. For .NET Framework Version 2.0 and 3.0, %VERSION% points to the netf20\ sub-directory.

/r:System.Data.dll
: Reference the Microsoft Windows System Data dynamic link library.

/r:System.dll
: Reference the Microsoft Windows System dynamic link library.

/r:System.Xml.dll
: Reference the Microsoft Windows System XML dynamic link library (for SubCOM.vb).

/r:SubCOM.dll
: Reference the SubCOM dynamic link library (for RootCOM.vb and LCTrans.vb).

/r:RootCOM.dll
: Reference the RootCOM dynamic link library (for LCTrans.vb).

Refer to your compiler documentation for additional compiler options.

C# .NET application compile and link options

This topic describes the various options available when compiling and linking C# .NET applications.

The compile and link options available to DB2 for building C# applications on Windows with the Microsoft C# compiler, as demonstrated in the bldapp.bat batch file.

Note: The .NET Framework Version 1.1 is supported only with the .NET Provider Version 9.5 and earlier.

Compile and link options for stand-alone C# applications using bldapp:

Compile and link options for stand-alone C# applications:

`%BLDCOMP%`
: Variable for the compiler. The default is `csc`, the Microsoft C# compiler.

`/r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll`
: Reference the DB2 dynamic link library for the .NET framework version you are using.

  `%VERSION%`
  : There are several supported versions of the .NET framework for applications. DB2 has a dynamic link library for each version. For .NET Framework Version 2.0, 3.0, and 3.5, `%VERSION%` points to the `netf20\` sub-directory.

Compile and link options for the loosely-coupled sample program, `LCTrans` using bldapp:

`%BLDCOMP%`
: Variable for the compiler. The default is `csc`, the Microsoft C# compiler.

`/out:RootCOM.dll`
: Output the `RootCOM` dynamic link library, used by the `LCTrans` application, from the `RootCOM.cs` source file,

`/out:SubCOM.dll`
: Output the `SubCOM` dynamic link library, used by the `LCTrans` application, from the `SubCOM.cs` source file,

`/target:library %1.cs`
: Create the dynamic link library from the input source file (`RootCOM.cs` or `SubCOM.cs`).

`/r:System.EnterpriseServices.dll`
: Reference the Microsoft Windows System EnterpriseServices data link library.

`/r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll`
: Reference the DB2 dynamic link library for the .NET framework version you are using.

  `%VERSION%`
  : There are several supported versions of the .NET framework for applications. DB2 has a dynamic link library for each. For .NET Framework Version 2.0, 3.0, and 3.5, `%VERSION%` points to the `netf20\` sub-directory.

`/r:System.Data.dll`
: Reference the Microsoft Windows System Data dynamic link library.

`/r:System.dll`
: Reference the Microsoft Windows System dynamic link library.

`/r:System.Xml.dll`
: Reference the Microsoft Windows System XML dynamic link library (for `SubCOM.cs`).

/r:SubCOM.dll
Reference the SubCOM dynamic link library (for `RootCOM.cs` and `LCTrans.cs`).

/r:RootCOM.dll
Reference the RootCOM dynamic link library (for `LCTrans.cs`).

Refer to your compiler documentation for additional compiler options.

OLE DB .NET Data Provider

The OLE DB .NET Data Provider uses the IBM DB2 OLE DB Driver, which is referred to in a `ConnectionString` object as `IBMDADB2`.

The connection string keywords supported by the OLE DB .NET Data Provider are the same as those supported by the IBM OLE DB Provider for DB2. This provider is no longer tested. Users are recommended to use the IBM Data Server Provider for .NET.

Also, the OLE DB .NET Data Provider has the same restrictions as the IBM DB2 OLE DB Provider. There are additional restrictions for the OLE DB .NET Data Provider, which are identified in the topic: "OLE DB .NET Data Provider restrictions" in *Developing ADO.NET and OLE DB Applications*.

In order to use the OLE DB .NET Data Provider, you must have the .NET Framework Version 2.0, 3.0, or 3.5 installed.

For DB2 Universal Database™ for AS/400® R520, R530 and R540, the following fix is required on the server: APAR ii13348.

All the supported connection keywords for the OLE DB .NET Data Provider are shown in table 1:

Table 3. Useful **ConnectionString** *keywords for the OLE DB .NET Data Provider*

Keyword	Value	Meaning
PROVIDER	IBMDADB2	Specifies the IBM OLE DB Provider for DB2 (required)
DSN or **Data Source**	database alias	The DB2 database alias as cataloged in the database directory
UID	user ID	The user ID used to connect to the DB2 data server
PWD	password	The password for the user ID used to connect to the DB2 data server

Note: For the full list of **ConnectionString** keywords, see the Microsoft documentation.

Example of creating an `OleDbConnection` to connect to the SAMPLE database is:

```
[Visual Basic .NET]
Dim con As New OleDbConnection("Provider=IBMDADB2;" +
    "Data Source=sample;UID=userid;PWD=password;")
con.Open()
```

```
[C#]
OleDbConnection con = new OleDbConnection("Provider=IBMDADB2;" +
                     "Data Source=sample;UID=userid;PWD=password;" );
con.Open()
```

OLE DB .NET Data Provider restrictions

The OLE DB .NET Data Provider is no longer tested. Users are recommended to use the IBM Data Server Provider for .NET.

The following table identifies usage restrictions for the OLE DB .NET Data Provider:

Table 4. OLE DB .NET Data Provider restrictions

Class or feature	Restriction description	DB2 servers affected
ASCII character streams	You cannot use ASCII character streams with `OleDbParameters` when using `DbType.AnsiString` or `DbType.AnsiStringFixedLength`. The OLE DB .NET Data Provider will throw the following exception: `"Specified cast is not valid"` **Workaround:** Use `DbType.Binary` instead of using `DbType.AnsiString` or `DbType.AnsiStringFixedLength`.	All
`ADORecord`	`ADORecord` is not supported.	All

`ADORecordSet` and `Timestamp`	As documented in MSDN, the `ADORecordSet` variant time resolves to one second. Consequently, all fractional seconds are lost when a DB2 `Timestamp` column is stored into a `ADORecordSet`. Similarly, after filling a `DataSet` from a `ADORecordSet`, the `Timestamp` columns in the `DataSet` will not have any fractional seconds.
	
	 Workaround: This workaround only works for DB2 Universal Database for Linux, UNIX, and Windows, Version 8.1, FixPak 4 or later. In order to avoid the loss of fraction of seconds, you can set the following CLI keyword:
	 `MAPTIMESTAMPDESCRIBE = 2`
	
	 This keyword will describe the `Timestamp` as a WCHAR(26). To set the keyword, execute the following command from a DB2 Command Window:
	 `db2 update cli cfg for section common using MAPTIMESTAMPDESCRIBE 2`
		All

`Chapters`	Chapters are not supported.	All
Key information	The OLE DB .NET Data Provider cannot retrieve key information when opening an `IDataReader` at the same time.	DB2 for VM/VSE

Table 4. OLE DB .NET Data Provider restrictions (continued)

Class or feature	Restriction description	DB2 servers affected
Key information from stored procedures	The OLE DB .NET Data Provider can retrieve key information about a result set returned by a stored procedure only from DB2 for Linux, UNIX, and Windows. This is because the DB2 servers for platforms other than Linux, UNIX, and Windows do not return extended describe information for the result sets opened in the stored procedure. In order to retrieve key information of a result set returned by a stored procedure on DB2 for Linux, UNIX, and Windows, you need to set the following registry variable on the DB2 server: `db2set DB2_APM_PERFORMANCE=8` Setting this server-side DB2 registry variable will keep the result set meta-data available on the server for a longer period of time, thus allowing OLE DB to successfully retrieve the key information. However, depending on the server workload, the meta-data might not be available long enough before the OLE DB Provider queries for the information. As such, there is no guarantee that the key information will always be available for result sets returned from a store procedure. In order to retrieve any key information about a CALL statement, the application must execute the CALL statement. Calling `OleDbDataAdapter.FillSchema()` or `OleDbCommand.ExecuteReader(CommandBehavior.SchemaOnly | CommandBehavior.KeyInfo)`, will not actually execute the stored procedure call. Therefore, you will not retrieve any key information for the result set that is to be returned by the stored procedure.	All

Key information from batched SQL statements	When using batched SQL statements that return multiple results, the `FillSchema()` method attempts to retrieve schema information only for the first SQL statement in the batched SQL statement list. If this statement does not return a result set then no table is created. For example:
	`[C#]`
	`cmd.CommandText =`
	`"INSERT INTO ORG(C1) VALUES(1000); SELECT C1 FROM ORG;";`
	`da = new OleDbDataAdapter(cmd);`
	`da.FillSchema(ds, SchemaType.Source);`
	
	No table will be created in the data set because the first statement in the batch SQL statement is an "INSERT" statement, which does not return a result set.
		All

Table 4. OLE DB .NET Data Provider restrictions (continued)

Class or feature	Restriction description	DB2 servers affected
`OleDbCommandBuilder`	The UPDATE, DELETE and INSERT statements automatically generated by the `OleDbCommandBuilder` are incorrect if the SELECT statement contains any columns of the following data types: • CLOB • BLOB • DBCLOB • LONG VARCHAR • LONG VARCHAR FOR BIT DATA • LONG VARGRAPHIC If you are connecting to a DB2 server other than DB2 for Linux, UNIX, and Windows, then columns of the following data types also cause this problem: • VARCHAR[1] • VARCHAR FOR BIT DATA[1] • VARGRAPHIC[1] • REAL • FLOAT or DOUBLE • TIMESTAMP **Note:** 1. Columns of these data types are applicable if they are defined to be VARCHAR values greater than 254 bytes, VARCHAR values FOR BIT DATA greater than 254 bytes, or VARGRAPHICs greater than 127 bytes. This condition is only valid if you are connecting to a DB2 server other than DB2 for Linux, UNIX, and Windows. The `OleDbCommandBuilder` generates SQL statements that use all of the selected columns in an equality comparison in the WHERE clause, but the data types listed previously cannot be used in an equality comparison. **Note:** Note that this restriction will affect the `IDbDataAdapter.Update()` method that relies on the `OleDbCommandBuilder` to automatically generate the UPDATE, DELETE, and INSERT statements. The UPDATE operation will fail if the generated statement contains any one of the data types listed previously. **Workaround:** You will need to explicitly remove all columns that are of the data types listed previously from the WHERE clause of the generated SQL statement. It is recommended that you code your own UPDATE, DELETE and INSERT statements.	All
`OleDbCommandBuilder.DeriveParameters`	Case-sensitivity is important when using `DeriveParameters()`. The stored procedure name specified in the `OleDbCommand.CommandText` needs to be in the same case as how it is stored in the DB2 system catalog tables. To see how stored procedure names are stored, call OpenSchema(OleDbSchemaGuid.Procedures) without supplying the procedure name restriction. This will return all the stored procedure names. By default, DB2 stores stored procedure names in uppercase, so most often, you need to specify the stored procedure name in uppercase.	All
`OleDbCommandBuilder.DeriveParameters`	The `OleDbCommandBuilder.DeriveParameters()` method does not include the `ReturnValue` parameter in the generated `OleDbParameterCollection`. SqlClient and the IBM Data Server Provider for .NET by default adds the parameter with `ParameterDirection.ReturnValue` to the generated `ParameterCollection`.	All
`OleDbCommandBuilder.DeriveParameters`	The `OleDbCommandBuilder.DeriveParameters()` method will fail for overloaded stored procedures. If you have multiple stored procedures of the name "MYPROC" with each of them taking a different number of parameters or different type of parameter, the `OleDbCommandBuilder.DeriveParameters()` will retrieve all the parameters for all the overloaded stored procedures.	All
`OleDbCommandBuilder.DeriveParameters`	If the application does not qualify a stored procedure with a schema, `DeriveParameters()` will return all the parameters for that procedure name. Therefore, if multiple schemas exist for the same procedure name, `DeriveParameters()` will return all the parameters for all the procedures with the same name.	All

Table 4. OLE DB .NET Data Provider restrictions (continued)

Class or feature	Restriction description	DB2 servers affected
`OleDbConnection.` `ChangeDatabase`	The `OleDbConnection.ChangeDatabase()` method is not supported.	All

`OleDbConnection.` `ConnectionString`	Use of nonprintable characters such as '\b', '\a' or '\O' in the connection string will cause an exception to be thrown.
	
	The following keywords have restrictions:
	
	`Data Source`
	The data source is the name of the database, not the server. You can specify the SERVER keyword, but it is ignored by the IBMDADB2 provider.
	
	`Initial Catalog` and `Connect Timeout`
	These keywords are not supported. In general, the OLE DB .NET Data Provider will ignore all unrecognized and unsupported keywords. However, specifying these keywords will cause the following exception:
	`Multiple-step OLE DB operation generated errors. Check each`
	`OLE DB status value, if available. No work was done.`
	
	`ConnectionTimeout`
	`ConnectionTimeout` is read only.
		All

`OleDbConnection.` `GetOleDbSchemaTable`	Restriction values are case-sensitive, and need to match the case of the database objects stored in the system catalog tables, which defaults to uppercase.
	
	For instance, if you have created a table in the following manner:
	`CREATE TABLE abc(c1 SMALLINT)`
	
	DB2 will store the table name in uppercase ("ABC") in the system catalog. Therefore, you will need to use "ABC" as the restriction value. For instance:
	`schemaTable = con.GetOleDbSchemaTable(OleDbSchemaGuid.Tables,`
	`        new object[] { null, null, "ABC", "TABLE" });`
	
	Workaround: If you need case-sensitivity or spaces in your data definitions, you must put quotation marks around them. For example:
	`cmd.CommandText = "create table \"Case Sensitive\"(c1 int)";`
	`cmd.ExecuteNonQuery();`
	`tablename = "\"Case Sensitive\"";`
	`schemaTable = con.GetOleDbSchemaTable(OleDbSchemaGuid.Tables,`
	`    new object[] { null, null, tablename, "TABLE" });`
		All

`OleDbDataAdapter` and `DataColumnMapping`	The source column name is case-sensitive. It needs to match the case as stored in the DB2 catalogs, which by default is uppercase. For example: `colMap = new DataColumnMapping("EMPNO", "Employee ID");`	All
`OleDbDataReader.` `GetSchemaTable`	The OLE DB .NET Data Provider is not able to retrieve extended describe information from servers that do not return extended describe information. if you are connecting to a server that does not support extended describe (the affected servers), the following columns in the metadata table returned from `IDataReader.GetSchemaTable()` are invalid: • `IsReadOnly` • `IsUnique` • `IsAutoIncrement` • `BaseSchemaName` • `BaseCatalogName`	DB2 for OS/390®, V7 or earlier DB2 for OS/400 DB2 for VM/VSE
Stored procedures: no column names for result sets	The DB2 for OS/390 version 6.1 server does not return column names for result sets returned from a stored procedure. The OLE DB .NET Data Provider maps these unnamed columns to their ordinal position (for example, "1", "2" "3"). This is contrary to the mapping documented in MSDN: `"Column1", "Column2", "Column3"`.	DB2 for OS/390 version 6.1

Hints and tips

Connection pooling in OLE DB .NET Data Provider applications

The OLE DB .NET Data Provider automatically pools connections using OLE DB session pooling.

Connection string arguments can be used to enable or disable OLE DB services including pooling. For example, the following connection string will disable OLE DB session pooling and automatic transaction enlistment.

```
Provider=IBMDADB2;OLE DB Services=-4;Data Source=SAMPLE;
```

The following table describes the ADO connection string attributes you can use to set the OLE DB services:

Table 5. Setting OLE DB services by using ADO connection string attributes

Services enabled	Value in connection string
All services (the default)	"OLE DB Services = -1;"
All services except pooling	"OLE DB Services = -2;"
All services except pooling and auto-enlistment	"OLE DB Services = -4;"
All services except client cursor	"OLE DB Services = -5;"
All services except client cursor and pooling	"OLE DB Services = -6;"
No services	"OLE DB Services = 0;"

For more information about OLE DB session pooling or resource pooling, as well as how to disable pooling by overriding OLE DB provider service defaults, see the OLE DB Programmer's Reference in the MSDN library located at:

```
http://msdn.microsoft.com/library
```

Time columns in OLE DB .NET Data Provider applications

You can insert data in time columns by binding time values to parameter markers. After you add the time values, you can retrieve the data using either the `IDataRecord.GetValue()` method or the `OleDbDataReader.GetTimeSpan()` method.

Inserting using parameter markers

You want to insert a time value into a Time column:

```
command.CommandText = "insert into mytable(c1) values( ? )";
```

where column c1 is a Time column. Here are two methods to bind a time value to the parameter marker:

Using `OleDbParameter.OleDbType = OleDbType.DBTime`

Because OleDbType.DBTime maps to a TimeSpan object, you must supply a TimeSpan object as the parameter value. The parameter value cannot be a String or a DateTime object, it must be a TimeSpan object. For example:

```
      p1.OleDbType = OleDbType.DBTime;
      p1.Value = TimeSpan.Parse("0.11:20:30");
      rowsAffected = cmd.ExecuteNonQuery();
```

The format of the TimeSpan is represented as a string in the format "[-]d.hh:mm:ss.ff" as documented in the MSDN documentation.

Using `OleDbParameter.OleDbType = OleDbType.DateTime`

This will force the OLE DB .NET Data Provider to convert the parameter value to a DateTime object, instead of a TimeSpan object, consequently the parameter value can be any valid string/object that can be converted into a DateTime object. This means values such as "11:20:30" will work. The value can also be a DateTime object. The value cannot be a TimeSpan object since a TimeSpan object cannot be converted to a DateTime object -- TimeSpan doesn't implement IConvertible.

For example:

```
p1.OleDbType = OleDbType.DBTimeStamp;
p1.Value = "11:20:30";
rowsAffected = cmd.ExecuteNonQuery();
```

Retrieval

To retrieve a time column you need to use the `IDataRecord.GetValue()` method or the `OleDbDataReader.GetTimeSpan()` method.

For example:

```
TimeSpan ts1 = ((OleDbDataReader)reader).GetTimeSpan( 0 );
TimeSpan ts2 = (TimeSpan) reader.GetValue( 0 );
```

ADORecordset objects in OLE DB .NET Data Provider applications

When you use ADORecordset objects, you must know which framework is being used and how each object you use maps to an ADORecordset object.

Considerations regarding the use of `ADORecordset` objects.

- The ADO type `adDBTime` class is mapped to the .NET Framework `DateTime` class. `OleDbType.DBTime` corresponds to a `TimeSpan` object.
- You cannot assign a `TimeSpan` object to an `ADORecordset` object's `Time` field. This is because the `ADORecordset` object's `Time` field expects a `DateTime` object. When you assign a `TimeSpan` object to an `ADORecordset` object, you will get the following message:

  ```
  Method's type signature is not Interop compatible.
  ```

  You can only populate the `Time` field with a `DateTime` object, or a `String` that can be parsed into a `DateTime` object.
- When you fill a `DataSet` with a `ADORecordset` using the `OleDbDataAdapter`, the `Time` field in the `ADORecordset` is converted to a `TimeSpan` column in the `DataSet`.
- `Recordsets` do not store primary keys or constraints. Therefore, no key information is added when filling out a `DataSet` from a `Recordset` using the `MissingSchemaAction.AddWithKey`.

ODBC .NET Data Provider

The ODBC .NET Data Provider makes ODBC calls to a DB2 data source using the CLI Driver. Therefore, the connection string keywords supported by the ODBC .NET Data Provider are the same as those supported by the CLI driver. This provider is no longer tested. Users are recommended to use the IBM Data Server Provider for .NET.

Also, the ODBC .NET Data Provider has the same restrictions as the CLI driver. There are additional restrictions for the ODBC .NET Data Provider, which are identified in the topic: "ODBC .NET Data Provider restrictions" in *Developing ADO.NET and OLE DB Applications*.

In order to use the ODBC .NET Data Provider, you must have the .NET Framework Version 2.0, 3.0, or 3.5 installed. For DB2 Universal Database for AS/400 V5R4 and earlier, the following fix is required on the server: APAR II13348.

The supported connection keywords for the ODBC .NET Data Provider are listed in the table 1:

Table 6. Useful **ConnectionString** *keywords for the ODBC .NET Data Provider*

Keyword	Value	Meaning
DSN	database alias	The DB2 database alias as cataloged in the database directory
UID	user ID	The user ID used to connect to the DB2 server
PWD	password	The password for the user ID used to connect to the DB2 server

Note: For the full list of **ConnectionString** keywords, see the Microsoft documentation.

The following code is an example of creating an OdbcConnection to connect to the SAMPLE database:

```
[Visual Basic .NET]
Dim con As New OdbcConnection("DSN=sample;UID=userid;PWD=password;")
con.Open()

[C#]
OdbcConnection con = new OdbcConnection("DSN=sample;UID=userid;PWD=password;");
con.Open()
```

ODBC .NET Data Provider restrictions

The ODBC .NET Data Provider is no longer tested. Users are recommended to use the IBM Data Server Provider for .NET.

The following table identifies usage restrictions for the ODBC .NET Data Provider:

Table 7. ODBC .NET Data Provider restrictions

Class or feature	Restriction description	DB2 servers affected
ASCII character streams	You cannot use ASCII character streams with OdbcParameters when using DbType.AnsiString or DbType.AnsiStringFixedLength. The ODBC .NET Data Provider will throw the following exception: "Specified cast is not valid" **Workaround:** Use DbType.Binary instead of using DbType.AnsiString or DbType.AnsiStringFixedLength.	All

Table 7. ODBC .NET Data Provider restrictions (continued)

Class or feature	**Restriction description**	**DB2 servers affected**
`Command.Prepare`	Before executing a command (`Command.ExecuteNonQuery` or `Command.ExecuteReader`), you must explicitly run `OdbcCommand.Prepare()` if the `CommandText` has changed since the last prepare. If you do not call `OdbcCommand.Prepare()` again, the ODBC .NET Data Provider will execute the previously prepared `CommandText`. For Example: `[C#]` `command.CommandText="select CLOB('ABC') from table1";` `command.Prepare();` `command.ExecuteReader();` `command.CommandText="select CLOB('XYZ') from table2";` `// This ends up re-executing the first statement` `command.ExecuteReader();`	All
CommandBehavior. SequentialAccess	When using `IDataReader.GetChars()` to read from a reader created with `CommandBehavior.SequentialAccess`, you must allocate a buffer that is large enough to hold the entire column. Otherwise, you will hit the following exception: `Requested range extends past the end of the array.` `at System.Runtime.InteropServices.Marshal.Copy(Int32 source, Char[] destination, Int32 startIndex, Int32 length)` `at System.Data.Odbc.OdbcDataReader.GetChars(Int32 i, Int64 dataIndex, Char[] buffer, Int32 bufferIndex, Int32 length)` `at OleRestrict.TestGetCharsAndBufferSize(IDbConnection con)` The following example demonstrates how to allocate an adequate buffer: `CREATE TABLE myTable(c0 int, c1 CLOB(10K))` `SELECT c1 FROM myTable;` `[C#]` `cmd.CommandText = "SELECT c1 from myTable";` `IDataReader reader =` `cmd.ExecuteReader(CommandBehavior.SequentialAccess);` `Int32 iChunkSize = 10;` `Int32 iBufferSize = 10;` `Int32 iFieldOffset = 0;` `Char[] buffer = new Char[ iBufferSize ];` `reader.Read();` `reader.GetChars(0, iFieldOffset, buffer, 0, iChunkSize);` The call to `GetChars()` will throw the following exception: `"Requested range extends past the end of the array"` To ensure that `GetChars()` does not throw the exception mentioned previously, you must set the `BufferSize` to the size of the column, as follows: `Int32 iBufferSize = 10000;` Note that the value of 10,000 for `iBufferSize` corresponds to the value of 10K allocated to the CLOB column `c1`.	All
CommandBehavior. SequentialAccess	The ODBC .NET Data Provider throws the following exception when there is no more data to read when using `OdbcDataReader.GetChars()`: `NO_DATA - no error information available` `at System.Data.Odbc.OdbcConnection.HandleError( HandleRef hrHandle, SQL_HANDLE hType, RETCODE retcode)` `at System.Data.Odbc.OdbcDataReader.GetData( Int32 i, SQL_C sqlctype, Int32 cb)` `at System.Data.Odbc.OdbcDataReader.GetChars( Int32 i, Int64 dataIndex, Char[] buffer, Int32 bufferIndex, Int32 length)`	All

Table 7. ODBC .NET Data Provider restrictions (continued)

Class or feature	Restriction description	DB2 servers affected

CommandBehavior. SequentialAccess	You may not be able to use large chunksizes, such as a value of 5000, when using `OdbcDataReader.GetChars()`. When you attempt to use a large chunk size, the ODBC .NET Data Provider will throw the following exception:
	`Object reference not set to an instance of an object.`
	`   at System.Runtime.InteropServices.Marshal.Copy(Int32 source,`
	`     Char[] destination, Int32 startIndex, Int32 length)`
	`   at System.Data.Odbc.OdbcDataReader.GetChars(`
	`    Int32 i, Int64 dataIndex, Char[] buffer,`
	`    Int32 bufferIndex, Int32 length)`
	`   at OleRestrict.TestGetCharsAndBufferSize(IDbConnection con)`
		All

Connection pooling	The ODBC .NET Data Provider does not control connection pooling. Connection pooling is handled by the ODBC Driver Manager. For more information about connection pooling, see the ODBC Programmer's Reference in the MSDN library located at `http://msdn.microsoft.com/library`	All
DataColumnMapping	The case of the source column name needs to match the case used in the system catalog tables, which is upper-case by default.	All

Decimal columns	Parameter markers are not supported for Decimal columns.
	
	You generally use `OdbcType.Decimal` for an `OdbcParameter` if the target `SQLType` is a `Decimal` column; however, when the ODBC .NET Data Provider sees the `OdbcType.Decimal`, it binds the parameter using C-type of `SQL_C_WCHAR` and `SQLType` of `SQL_VARCHAR`, which is invalid.
	
	For example:
	`[C#]`
	`cmd.CommandText = "SELECT dec_col FROM MYTABLE WHERE dec_col > ? ";`
	`OdbcParameter p1 = cmd.CreateParameter();`
	`p1.DbType = DbType.Decimal;`
	`p1.Value = 10.0;`
	`cmd.Parameters.Add(p1);`
	`IDataReader rdr = cmd.ExecuteReader();`
	
	You will get an exception:
	`ERROR [07006] [IBM][CLI Driver][SQLDS/VM]`
	`   SQL0301N  The value of input host variable or parameter`
	`   number "" cannot be used because of its data type.`
	`    SQLSTATE=07006`
	
	Workaround: Instead of using `OdbcParameter` values, use literals exclusively.
		DB2 for VM/VSE

Key information	The schema name used to qualify the table name (for example, `MYSCHEMA.MYTABLE`) must match the connection user ID. The ODBC .NET Data Provider is unable to retrieve any key information in which the specified schema is different from the connection user id.
	
	For example:
	`CREATE TABLE USERID2.TABLE1(c1 INT NOT NULL PRIMARY KEY);`
	
	`[C#]`
	`// Connect as user bob`
	`odbcCon = new OdbcConnection("DSN=sample;UID=bob;PWD=mypassword");`
	
	`OdbcCommand cmd = odbcCon.CreateCommand();`
	
	`// Select from table with schema USERID2`
	`cmd.CommandText="SELECT * FROM USERID2.TABLE1";`
	
	`// Fails - No key info retrieved`
	`da.FillSchema(ds, SchemaType.Source);`
	
	`// Fails - SchemaTable has no primary key`
	`cmd.ExecuteReader(CommandBehavior.KeyInfo)`
	
	`// Throws exception because no primary key`
	`cbuilder.GetUpdateCommand();`
		All

Table 7. ODBC .NET Data Provider restrictions (continued)

Class or feature	**Restriction description**	**DB2 servers affected**

Key information	The ODBC .NET Data Provider cannot retrieve key information when opening a `IDataReader` at the same time. When the ODBC .NET Data Provider opens a `IDataReader`, a cursor on the server is opened. If key information is requested, it will then call `SQLPrimaryKeys()` or `SQLStatistic()` to get the key information, but these schema functions will open another cursor. Since DB2 for VM/VSE does not support cursor withhold, the first cursor is then closed. Consequently, `IDataReader.Read()` calls to the `IDataReader` will result in the following exception:
	`System.Data.Odbc.OdbcException: ERROR [HY010] [IBM][CLI Driver]`
	`  CLI0125E  Function sequence error. SQLSTATE=HY010`
	
	Workaround: You will need to retrieve key information first then retrieve the data. For example:
	`[C#]`
	`OdbcCommand cmd = odbcCon.CreateCommand();`
	`OdbcDataAdapter da = new OdbcDataAdapter(cmd);`
	
	`cmd.CommandText  = "SELECT * FROM MYTABLE";`
	
	`// Use FillSchema to retrieve just the schema information`
	`da.FillSchema(ds, SchemaType.Source);`
	`// Use FillSchema to retrieve just the schema information`
	`da.Fill(ds);`
		DB2 for VM/VSE

Key information	You must refer to database objects in your SQL statements using the same case that the database objects are stored in the system catalog tables. By default database objects are stored in uppercase in the system catalog tables, so most often, you need to use uppercase. The ODBC .NET Data Provider scans SQL statements to retrieve database object names and passes them to schema functions such as `SQLPrimaryKeys` and `SQLStatistics`, which issue queries for these objects in the system catalog tables. The database object references must match exactly how they are stored in the system catalog tables, otherwise, an empty result set is returned.	DB2 for OS/390 DB2 for OS/400 DB2 for VM/VSE
Key information for batched non-select SQL statements	The ODBC .NET Data Provider is unable to retrieve any key information for a batch statement that does not start with "SELECT".	DB2 for OS/390 DB2 for OS/400 DB2 for VM/VSE

Table 7. ODBC .NET Data Provider restrictions (continued)

Class or feature	Restriction description	DB2 servers affected

LOB columns	The ODBC .NET Data Provider does not support LOB datatypes. Consequently, whenever the DB2 server returns a SQL_CLOB (-99), SQL_BLOB (-98) or SQL_DBCLOB (-350) the ODBC .NET Data Provider will throw the following exception:
	`"Unknown SQL type - -98"     (for Blob column)`
	`"Unknown SQL type - -99"     (for Clob column)`
	`"Unknown SQL type - -350"    (for DbClob column)`
	
	Any methods that directly or indirectly access LOB columns will fail.
	
	Workaround: Set the CLI/ODBC `LongDataCompat` keyword to 1. Doing so will force the CLI driver to make the following data type mappings to data types the ODBC .NET Data Provider will understand:
	• SQL_CLOB to SQL_LONGVARCHAR
	• SQL_BLOB to SQL_LONGVARBINARY
	• SQL_DBCLOB to SQL_WLONGVARCHAR
	
	To set the `LongDataCompat` keyword, run the following DB2 command from a DB2 command window on the client machine:
	`db2 update cli cfg for section common using longdatacompat 1`
	
	You can also set this keyword in your application, using the connection string as follows:
	`[C#]`
	`OdbcConnection con =`
	`  new OdbcConnection("DSN=SAMPLE;UID=uid;PWD=mypwd;LONGDATACOMPAT=1;");`
	
	For a list of all the CLI/ODBC keywords, refer to the "UID CLI/ODBC configuration keyword" in the DB2 CLI Guide and Reference.
		All

`OdbcCommand.Cancel`	Executing statements after running `OdbcCommand.Cancel` can lead to the following exception:
	`"ERROR [24000] [Microsoft][ODBC Driver Manager]`
	`    Invalid cursor state"`
		All

`OdbcCommandBuilder`	The `OdbcCommandBuilder` fails to generate commands against servers that do not support escape characters. When the `OdbcCommandBuilder` generates commands, it first makes a call to `SQLGetInfo`, requesting the `SQL_SEARCH_PATTERN_ESCAPE` attribute. If the server does not support escape characters an empty string is returned, which causes the ODBC .NET Data Provider to throw the following exception:
	`Index was outside the bounds of the array.`
	`   at System.Data.Odbc.OdbcConnection.get_EscapeChar()`
	`   at System.Data.Odbc.OdbcDataReader.GetTableNameFromCommandText()`
	`   at System.Data.Odbc.OdbcDataReader.BuildMetaDataInfo()`
	`   at System.Data.Odbc.OdbcDataReader.GetSchemaTable()`
	`   at System.Data.Common.CommandBuilder.BuildCache(`
	`    Boolean closeConnection)`
	`   at System.Data.Odbc.OdbcCommandBuilder.GetUpdateCommand()`
		DB2 for OS/390, DBCS servers only; DB2 for VM/VSE, DBCS servers only

Table 7. ODBC .NET Data Provider restrictions (continued)

Class or feature	**Restriction description**	**DB2 servers affected**

`OdbcCommandBuilder`	Case-sensitivity is important when using the `OdbcCommandBuilder` to automatically generate UPDATE, DELETE, and INSERT statements. By default, DB2 stores schema information (such as table names, and column names) in the system catalog tables in upper case, unless they have been explicitly created with case-sensitivity (by adding quotation marks around database objects during create-time). As such, your SQL statements must match the case that is stored in the catalogs (which by default is uppercase).
	
	For example, if you created a table using the following statement:
	`"db2 create table mytable (c1 int) "`
	
	then DB2 will store the table name "mytable" in the system catalog tables as "MYTABLE".
	
	The following code example demonstrates proper use the `OdbcCommandBuilder`class:
	`[C#]`
	`OdbcCommand cmd = odbcCon.CreateCommand();`
	`cmd.CommandText  = "SELECT * FROM MYTABLE";`
	`OdbcDataAdapter da = new OdbcDataAdapter(cmd);`
	`OdbcCommandBuilder cb = new OdbcCommandBuilder(da);`
	`OdbcCommand updateCmd = cb.GetUpdateCommand();`
	
	In this example, if you do not refer to the table name in upper-case characters, then you will get the following exception:
	`"Dynamic SQL generation for the UpdateCommand is not`
	`supported against a SelectCommand that does not return`
	`any key column information."`
		All

OdbcCommandBuilder	The commands generated by the `OdbcCommandBuilder` are incorrect when the SELECT statement contains the following column data types:
	`REAL`
	`FLOAT or DOUBLE`
	`TIMESTAMP`
	
	These data types cannot be used in the WHERE clause for SELECT statements.
		DB2 for OS/390
		DB2 for OS/400
		DB2 for VM/VSE

OdbcCommandBuilder. DeriveParameters	The `DeriveParameters()` method is mapped to `SQLProcedureColumns` and it uses the `CommandText` property for the name of the stored procedure. Since `CommandText` does not contain the name of the stored procedure (using full ODBC call syntax), `SQLProcedureColumns` is called with the procedure name identified according to the ODBC call syntax. For example: `"{ CALL myProc(?) }"` This which will result in an empty result set, where no columns are found for the procedure).	All
OdbcCommandBuilder. DeriveParameters	To use `DeriveParameters()`, specify the stored procedure name in the `CommandText` (for example, `cmd.CommandText = "MYPROC"`). The procedure name must match the case stored in the system catalog tables. `DeriveParameters()` will return all the parameters for that procedure name it finds in the system catalog tables. Remember to change the `CommandText` back to the full ODBC call syntax before executing the statement.	All
OdbcCommandBuilder. DeriveParameters	The `ReturnValue` parameter is not returned for the ODBC .NET Data Provider.	All
OdbcCommandBuilder. DeriveParameters	`DeriveParameters()` does not support fully qualified stored procedure names. For example, calling `DeriveParameters()` for `CommandText = "MYSCHEMA.MYPROC"` will fail. Here, no parameters are returned.	All
OdbcCommandBuilder. DeriveParameters	`DeriveParameters()` will not work for overloaded stored procedures. The `SQLProcedureColumns` will return all the parameters for all versions of the stored procedure.	All
`OdbcConnection.` `ChangeDatabase`	The `OdbcConnection.ChangeDatabase()` method is not supported.	All

Table 7. ODBC .NET Data Provider restrictions (continued)

Class or feature	Restriction description	DB2 servers affected
`OdbcConnection.` `ConnectionString`	• The `Server` keyword is ignored. • The `Connect Timeout` keyword is ignored. CLI does not support connection timeouts, so setting this property will not affect the driver. • Connection pooling keywords are ignored. Specifically, this affects the following keywords: `Pooling`, `Min Pool Size`, `Max Pool Size`, `Connection Lifetime` and `Connection Reset`.	All
OdbcDataReader. GetSchemaTable	The ODBC .NET Data Provider is not able to retrieve extended describe information from servers that do not return extended describe information. Therefore, if you are connecting to a server that does not support extended describe (the affected servers), the following columns in the metadata table returned from `IDataReader.GetSchemaTable()` are invalid: • `IsReadOnly` • `IsUnique` • `IsAutoIncrement` • `BaseSchemaName` • `BaseCatalogName`	DB2 for OS/390, version 7 or lower DB2 for OS/400 DB2 for VM/VSE

Stored procedures	To call a stored procedure, you need to specify the full ODBC call syntax.
	
	For example, to call the stored procedure, `MYPROC`, that takes a VARCHAR(10) as a parameter:
	`[C#]`
	`OdbcCommand cmd = odbcCon.CreateCommand();`
	`cmd.CommandType = CommandType.Text;`
	`cmd.CommandText = "{ CALL MYPROC(?) }"`
	`OdbcParameter p1 = cmd.CreateParameter();`
	`p1.Value = "Joe";`
	`p1.OdbcType = OdbcType.NVarChar;`
	`cmd.Parameters.Add(p1);`
	`cmd.ExecuteNonQuery();`
	
	Note: Note that you must use the full ODBC call syntax even if you are using `CommandType.StoredProcedure`. This is documented in MSDN, under the OdbcCommand.CommandText Property.
		All

Stored procedures: no column names for result sets	The DB2 for OS/390 version 6.1 server does not return column names for result sets returned from a stored procedure. The ODBC .NET Data Provider maps these unnamed columns to their ordinal position (for example, "1", "2" "3"). This is contrary to the mapping documented in MSDN: `"Column1"`, `"Column2"`, `"Column3"`.	DB2 for OS/390 version 6.1
Unique index promotion to primary key	The ODBC .NET Data Provider promotes nullable unique indexes to primary keys. This is contrary to the MSDN documentation, which states that nullable unique indexes should not be promoted to primary keys.	All

Chapter 2. IBM OLE DB Provider for DB2

The IBM OLE DB Provider for DB2 allows DB2 to act as a resource manager for the OLE DB provider. This support gives OLE DB-based applications the ability to extract or query DB2 data using the OLE interface.

Microsoft OLE DB is a set of OLE/COM interfaces that provides applications with uniform access to data stored in diverse information sources. The OLE DB architecture defines OLE DB consumers and OLE DB providers. An OLE DB consumer is any system or application that uses OLE DB interfaces; an OLE DB provider is a component that exposes OLE DB interfaces.

The IBM OLE DB Provider for DB2, whose provider name is IBMDADB2, enables OLE DB consumers to access data on a DB2 database server. If DB2 Connect is installed, these OLE DB consumers can also access data on a host DBMS such as DB2 for z/OS, DB2 Server for VM and VSE, or DB2 Universal Database for AS/400.

The IBM OLE DB Provider for DB2 offers the following features:

- Support level 0 of the OLE DB provider specification, including some additional level 1 interfaces.
- A free threaded provider implementation, which enables the application to create components in one thread and use those components in any other thread.
- An Error Lookup Service that returns DB2 error messages.

Note that the IBM OLE DB Provider resides on the client and is different from the OLE DB table functions, which are also supported by DB2 database systems.

Subsequent sections of this document describe the specific implementation of the IBM OLE DB Provider for DB2. For more information about the Microsoft OLE DB 2.0 specification, refer to the Microsoft OLE DB 2.0 Programmer's Reference and Data Access SDK, available from Microsoft Press.

Version Compliance

The IBM OLE DB Provider for DB2 complies with Version 2.7 or later of the Microsoft OLE DB specification.

System Requirements

Refer to the announcement letter for the IBM OLE DB Provider for DB2 data servers to see the supported Windows operating systems.

To install the IBM OLE DB Provider for DB2, you must first be running on one of the supported operating systems listed previously. You also need to install a full DB2 product, IBM Data Server Driver for ODBC and CLI, or IBM Data Server Driver Package.

Application Types Supported by the IBM OLE DB Provider for DB2

Not all application types are supported by the IBM OLE DB Provider for DB2 databases. When you are designing your application, you must ensure that the type you have chosen is supported.

With the IBM OLE DB Provider for DB2, you can create the following types of applications:

- ADO applications, including:
  - Microsoft Visual Studio C++ applications
  - Microsoft Visual Basic applications
- ADO.NET applications using the OLE DB .NET Data Provider
- C/C++ applications which access IBMDADB2 directly using the OLE DB interfaces, including ATL applications whose Data Access Consumer Objects were generated by the ATL COM AppWizard.

OLE DB services

Thread model supported by the IBM OLE DB Provider

The IBM OLE DB Provider for DB2 databases supports the Free Threaded model. You can use the Free Threaded model to write applications that create components in one thread and use those components in another thread.

Large object manipulation with the IBM OLE DB Provider

You can get and set data as storage objects (`DBTYPE_IUNKNOWN`) with the IBMDADB2 provider by using the ISequentialStream interface.

You can use the ISequentialStream interface in the following ways:

- To bind a storage object to a parameter, the DBOBJECT in the DBBINDING structure can only contain the value `STGM_READ` for the dwFlag field. IBMDADB2 will execute the Read method of the ISequentialStream interface of the bound object.
- To get data from a storage object, your application must run the Read method on the ISequentialStream interface of the storage object.
- When getting data, the value of the length part is the length of the real data, not the length of the IUnknown pointer.

Schema rowsets supported by the IBM OLE DB Provider

The following table shows the schema rowsets that are supported by `IDBSchemaRowset`. Unsupported columns will be set to `null` in the rowsets.

Table 8. Schema Rowsets Supported by the IBM OLE DB Provider for DB2

Supported GUIDs	Supported Restrictions	Supported Columns	Notes
DBSCHEMA _COLUMN_PRIVILEGES	COLUMN_NAME TABLE_NAME TABLE_SCHEMA	COLUMN_NAME GRANTEE GRANTOR IS_GRANTABLE PRIVILEGE_TYPE TABLE_NAME TABLE_SCHEMA	

Table 8. Schema Rowsets Supported by the IBM OLE DB Provider for DB2 (continued)

Supported GUIDs	Supported Restrictions	Supported Columns	Notes
DBSCHEMA_COLUMNS	COLUMN_NAME TABLE_NAME TABLE_SCHEMA	CHARACTER_MAXIMUM_LENGTH CHARACTER_OCTET_LENGTH COLUMN_DEFAULT COLUMN_FLAGS COLUMN_HASDEFAULT COLUMN_NAME DATA_TYPE DESCRIPTION IS_NULLABLE NUMERIC_PRECISION NUMERIC_SCALE ORDINAL_POSITION TABLE_NAME TABLE_SCHEMA	
DBSCHEMA_FOREIGN_KEYS	FK_TABLE_NAME FK_TABLE_SCHEMA PK_TABLE_NAME PK_TABLE_SCHEMA	DEFERRABILITY DELETE_RULE FK_COLUMN_NAME FK_NAME FK_TABLE_NAME FK_TABLE_SCHEMA ORDINAL PK_COLUMN_NAME PK_NAME PK_TABLE_NAME PK_TABLE_SCHEMA UPDATE_RULE	Must specify at least one of the following restrictions: PK_TABLE_NAME or FK_TABLE_NAME No "%" wildcard allowed.
DBSCHEMA_INDEXES	TABLE_NAME TABLE_SCHEMA	CARDINALITY CLUSTERED COLLATION COLUMN_NAME INDEX_NAME INDEX_SCHEMA ORDINAL_POSITION PAGES TABLE_NAME TABLE_SCHEMA TYPE UNIQUE	No sort order supported. Sort order, if specified, will be ignored.
DBSCHEMA_PRIMARY_KEYS	TABLE_NAME TABLE_SCHEMA	COLUMN_NAME ORDINAL PK_NAME TABLE_NAME TABLE_SCHEMA	Must specify at least the following restrictions: TABLE_NAME No "%" wildcard allowed.

Table 8. Schema Rowsets Supported by the IBM OLE DB Provider for DB2 (continued)

Supported GUIDs	**Supported Restrictions**	**Supported Columns**	**Notes**
DBSCHEMA _PROCEDURE_PARAMETERS	PARAMETER_NAME PROCEDURE_NAME PROCEDURE_SCHEMA	CHARACTER_MAXIMUM_LENGTH CHARACTER_OCTET_LENGTH DATA_TYPE DESCRIPTION IS_NULLABLE NUMERIC_PRECISION NUMERIC_SCALE ORDINAL_POSITION PARAMETER_DEFAULT PARAMETER_HASDEFAULT PARAMETER_NAME PARAMETER_TYPE PROCEDURE_NAME PROCEDURE_SCHEMA TYPE_NAME	
DBSCHEMA_PROCEDURES	PROCEDURE_NAME PROCEDURE_SCHEMA	DESCRIPTION PROCEDURE_NAME PROCEDURE_SCHEMA PROCEDURE_TYPE	
DBSCHEMA_PROVIDER_TYPES	DATA_TYPE BEST_MATCH	AUTO_UNIQUE_VALUE BEST_MATCH CASE_SENSITIVE CREATE_PARAMS COLUMN_SIZE DATA_TYPE FIXED_PREC_SCALE IS_FIXEDLENGTH IS_LONG IS_NULLABLE LITERAL_PREFIX LITERAL_SUFFIX LOCAL_TYPE_NAME MINIMUM_SCALE MAXIMUM_SCALE SEARCHABLE TYPE_NAME UNSIGNED_ATTRIBUTE	
DBSCHEMA_STATISTICS	TABLE_NAME TABLE_SCHEMA	CARDINALITY TABLE_NAME TABLE_SCHEMA	No sort order supported. Sort order, if specified, will be ignored.
DBSCHEMA _TABLE_PRIVILEGES	TABLE_NAME TABLE_SCHEMA	GRANTEE GRANTOR IS_GRANTABLE PRIVILEGE_TYPE TABLE_NAME TABLE_SCHEMA	
DBSCHEMA_TABLES	TABLE_NAME TABLE_SCHEMA TABLE_TYPE	DESCRIPTION TABLE_NAME TABLE_SCHEMA TABLE_TYPE	

OLE DB services automatically enabled by the IBM OLE DB Provider

By default, the IBM OLE DB Provider for DB2 databases automatically enables all the OLE DB services by adding a registry entry `OLEDB_SERVICES` under the class ID (CLSID) of the provider with the DWORD value of `0xFFFFFFFF`.

The values of the DWORD value determine the number and type of OLE DB services available.

Table 9. OLE DB Services

Enabled Services	DWORD Value
All services (default)	`0xFFFFFFFF`
All except pooling and AutoEnlistment	`0xFFFFFFFC`
All except client cursor	`0xFFFFFFFB`
All except pooling, enlistment and cursor	`0xFFFFFFF8`
No services	`0x000000000`

Data services

Supported cursor modes for the IBM OLE DB Provider

The IBM OLE DB Provider for DB2 databases natively supports read-only, forward-only, updatable scrollable, and updatable scrollable cursors.

Data type mappings between DB2 and OLE DB

The IBM OLE DB Provider for DB2 supports data type mappings between DB2 data types and OLE DB data types.

The following table provides a complete list of supported mappings and available names for indicating the data types of columns and parameters.

Table 10. Data type mappings between DB2 data types and OLE DB data types

DB2 Data Types	OLE DB Data Types Indicators	OLE DB Standard Type Names	DB2 Specific Names
SMALLINT	DBTYPE_I2	"DBTYPE_I2"	"SMALLINT"
INTEGER	DBTYPE_I4	"DBTYPE_I4"	"INTEGER" or "INT"
BIGINT	DBTYPE_I8	"DBTYPE_I8"	"BIGINT"
REAL	DBTYPE_R4	"DBTYPE_R4"	"REAL"
FLOAT	DBTYPE_R8	"DBTYPE_R8"	"FLOAT"
DOUBLE	DBTYPE_R8	"DBTYPE_R8"	"DOUBLE" or "DOUBLE PRECISION"
DECIMAL	DBTYPE_NUMERIC	"DBTYPE_NUMERIC"	"DEC" or "DECIMAL"
NUMERIC	DBTYPE_NUMERIC	"DBTYPE_NUMERIC"	"NUM" or "NUMERIC"
DATE	DBTYPE_DBDATE	"DBTYPE_DBDATE"	"DATE"
TIME	DBTYPE_DBTIME	"DBTYPE_DBTIME"	"TIME"
TIMESTAMP	DBTYPE_DBTIMESTAMP	"DBTYPE_DBTIMESTAMP"	"TIMESTAMP"

Table 10. Data type mappings between DB2 data types and OLE DB data types (continued)

DB2 Data Types	OLE DB Data Types Indicators	OLE DB Standard Type Names	DB2 Specific Names
CHAR	DBTYPE_STR	"DBTYPE_CHAR"	"CHAR" or "CHARACTER"
VARCHAR	DBTYPE_STR	"DBTYPE_VARCHAR"	"VARCHAR"
LONG VARCHAR	DBTYPE_STR	"DBTYPE_LONGVARCHAR"	"LONG VARCHAR"
CLOB	DBTYPE_STR and DBCOLUMNFLAGS_ISLONG or DBPARAMFLAGS_ISLONG	"DBTYPE_CHAR" "DBTYPE_VARCHAR" "DBTYPE_LONGVARCHAR" and DBCOLUMNFLAGS_ISLONG or DBPARAMFLAGS_ISLONG	"CLOB"
GRAPHIC	DBTYPE_WSTR	"DBTYPE_WCHAR"	"GRAPHIC"
VARGRAPHIC	DBTYPE_WSTR	"DBTYPE_WVARCHAR"	"VARGRAPHIC"
LONG VARGRAPHIC	DBTYPE_WSTR	"DBTYPE_WLONGVARCHAR"	"LONG VARGRAPHIC"
DBCLOB	DBTYPE_WSTR and DBCOLUMNFLAGS_ISLONG or DBPARAMFLAGS_ISLONG	"DBTYPE_WCHAR" "DBTYPE_WVARCHAR" "DBTYPE_WLONGVARCHAR" and DBCOLUMNFLAGS_ISLONG or DBPARAMFLAGS_ISLONG	"DBCLOB"
CHAR(n) FOR BIT DATA	DBTYPE_BYTES	"DBTYPE_BINARY"	
VARCHAR(n) FOR BIT DATA	DBTYPE_BYTES	"DBTYPE_VARBINARY"	
LONG VARCHAR FOR BIT DATA	DBTYPE_BYTES	"DBTYPE_LONGVARBINARY"	
BLOB	DBTYPE_BYTES and DBCOLUMNFLAGS_ISLONG or DBPARAMFLAGS_ISLONG	"DBTYPE_BINARY" "DBTYPE_VARBINARY" "DBTYPE_LONGVARBINARY" and DBCOLUMNFLAGS_ISLONG or DBPARAMFLAGS_ISLONG	"BLOB"

Data conversion for setting data from OLE DB Types to DB2 Types

The IBM OLE DB Provider for DB2 supports data conversions for setting data from OLE DB types to DB2 types.

Supported data conversion from OLE DB Types to DB2 Types

The following table shows data conversions from OLE DB types to DB2 types. Note that truncation of the data may occur in some cases, depending on the types and the value of the data.

Table 11. Data conversions from OLE DB types to DB2 types

	DB2 Data Types																					
																		For Bit Data				
OLE DB Type Indicator	SMALLINT	INTEGER	BIGINT	REAL	FLOAT DOUBLE	DECIMAL NUMERIC	DATE	TIME	TIMESTAMP	CHAR	VARCHAR	LONG VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	LONG VARGRAPHIC	DBCLOB	CHAR	VARCHAR	LONG VARCHAR	BLOB	DATA LINK
DBTYPE_EMPTY																						
DBTYPE_NULL																						
DBTYPE_RESERVED																						
DBTYPE_I1	X	X	X	X	X	X				X	X											
DBTYPE_I2	X	X	X	X	X	X				X	X											
DBTYPE_I4	X	X	X	X	X	X				X	X											
DBTYPE_I8	X	X	X	X	X	X				X	X											
DBTYPE_UI1	X	X	X	X	X	X				X	X											
DBTYPE_UI2	X	X	X	X	X	X				X	X											
DBTYPE_UI4	X	X	X	X	X	X				X	X											
DBTYPE_UI8	X	X	X	X	X	X				X	X											
DBTYPE_R4	X	X	X	X	X	X				X	X											
DBTYPE_R8	X	X	X	X	X	X				X	X											
DBTYPE_CY																						
DBTYPE_DECIMAL	X	X	X	X	X	X				X	X											
DBTYPE_NUMERIC	X	X	X	X	X	X				X	X											

Table 11. Data conversions from OLE DB types to DB2 types (continued)

	DB2 Data Types																					
																		For Bit Data				
OLE DB Type Indicator	**SMALLINT**	**INTEGER**	**BIGINT**	**REAL**	**FLOAT DOUBLE**	**DECIMAL NUMERIC**	**DATE**	**TIME**	**TIMESTAMP**	**CHAR**	**VARCHAR**	**LONG VARCHAR**	**CLOB**	**GRAPHIC**	**VARGRAPHIC**	**LONG VARGRAPHIC**	**DBCLOB**	**CHAR**	**VARCHAR**	**LONG VARCHAR**	**BLOB**	**DATALINK**
DBTYPE_DATE																						
DBTYPE_BOOL	X	X	X	X	X	X				X	X											
DBTYPE_BYTES			X			X				X	X	X				X		X	X	X		
DBTYPE_BSTR - to be determined																						
DBTYPE_STR	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_WSTR														X	X	X						
DBTYPE_VARIANT - to be determined																						
DBTYPE_IDISPATCH																						
DBTYPE_IUNKNOWN										X	X	X	X	X	X	X	X	X	X	X		
DBTYPE_GUID																						
DBTYPE_ERROR																						
DBTYPE_BYREF																						
DBTYPE_ARRAY																						
DBTYPE_VECTOR																						
DBTYPE_UDT																						

Table 11. Data conversions from OLE DB types to DB2 types (continued)

OLE DB Type Indicator	DB2 Data Types																					
	SMALLINT	INTEGER	BIGINT	REAL	FLOAT DOUBLE	DECIMAL NUMERIC	DATE	TIME	TIMESTAMP	CHAR	VARCHAR	LONG VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	LONG VARGRAPHIC	DBCLOB	For Bit Data			BLOB	DATALINK
																		CHAR	VARCHAR	LONG VARCHAR		
DBTYPE_DBDATE							X		X	X	X											
DBTYPE_DBTIME								X	X	X	X											
DBTYPE_DBTIMESTAMP							X	X	X	X	X											
DBTYPE_FILETIME																						
DBTYPE_PROP_VARIANT																						
DBTYPE_HCHAPTER																						
DBTYPE_VARNUMERIC																						

Data conversion for setting data from DB2 types to OLE DB types

For getting data, the IBM OLE DB Provider allows data conversions from DB2 types to OLE DB types.

Supported data conversions from DB2 types to OLE DB types

The following table shows supported data conversions from DB2 types to OLE DB types. Note that truncation of the data may occur in some cases, depending on the types and the value of the data.

Table 12. Data conversions from DB2 types to OLE DB types

	DB2 Data Types																					
																		For Bit Data				
OLE DB Type Indicator	**SMALLINT**	**INTEGER**	**BIGINT**	**REAL**	**FLOAT DOUBLE**	**DECIMAL NUMERIC**	**DATE**	**TIME**	**TIMESTAMP**	**CHAR**	**VARCHAR**	**LONG VARCHAR**	**CLOB**	**GRAPHIC**	**VARGRAPHIC**	**LONG VARGRAPHIC**	**DBCLOB**	**CHAR**	**VARCHAR**	**LONG VARCHAR**	**BLOB**	**DATALINK**
DBTYPE_EMPTY																						
DBTYPE_NULL																						
DBTYPE_RESERVED																						
DBTYPE_I1	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_I2	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_I4	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_I8	X	X	X	X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_UI1	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_UI2	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_UI4	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_UI8	X	X	X	X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_R4	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_R8	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_CY	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_DECIMAL	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_NUMERIC	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X

Table 12. Data conversions from DB2 types to OLE DB types (continued)

	DB2 Data Types																				
																	For Bit Data				
OLE DB Type Indicator	SMALLINT	INTEGER	BIGINT	REAL	FLOAT DOUBLE	DECIMAL NUMERIC	DATE	TIME	TIMESTAMP	CHAR	VARCHAR	LONG VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	LONG VARGRAPHIC	DBCLOB	CHAR	VARCHAR	LONG VARCHAR	BLOB	DATA LINK
DBTYPE_DATE	X	X		X	X		X	X	X	X	X	X		X	X	X						X
DBTYPE_BOOL	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_BYTES	X	X		X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_BSTR	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_STR	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_WSTR	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_VARIANT	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_IDISPATCH																						
DBTYPE_IUNKNOWN	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DBTYPE_GUID										X	X	X		X	X	X		X	X	X		X
DBTYPE_ERROR																						
DBTYPE_BYREF																						
DBTYPE_ARRAY																						
DBTYPE_VECTOR																						
DBTYPE_UDT																						
DBTYPE_DBDATE							X	X	X	X	X	X		X	X	X		X	X	X		X

Table 12. Data conversions from DB2 types to OLE DB types *(continued)*

	DB2 Data Types																					
																		For Bit Data				
OLE DB Type Indicator	**SMALLINT**	**INTEGER**	**BIGINT**	**REAL**	**FLOAT DOUBLE**	**DECIMAL NUMERIC**	**DATE**	**TIME**	**TIMESTAMP**	**CHAR**	**VARCHAR**	**LONG VARCHAR**	**CLOB**	**GRAPHIC**	**VARGRAPHIC**	**LONG VARGRAPHIC**	**DBCLOB**	**CHAR**	**VARCHAR**	**LONG VARCHAR**	**BLOB**	**DATA LINK**
DBTYPE_DBTIME							X	X	X	X	X	X		X	X	X						X
DBTYPE_DBTIMESTAMP							X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_FILETIME			X				X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_PROP_VARIANT	X	X	X	X	X					X	X	X		X	X	X		X	X	X		X
DBTYPE_HCHAPTER																						
DBTYPE_VARNUMERIC																						

Note: When the application performs the ISequentialStream::Read to get the data from the storage object, the format of the data returned depends on the column data type:

- For non character and binary data types, the data of the column is exposed as a sequence of bytes which represent those values in the operating system.
- For character data type, the data is first converted to DBTYPE_STR.
- For DBCLOB, the data is first converted to DBTYPE_WCHAR.

IBM OLE DB Provider restrictions

The restrictions for the IBM OLE DB Provider are:

- IBMDADB2 supports auto commit and user-controlled transaction scope with the `ITransactionLocal` interface. Auto commit transaction scope is the default scope. Nested transactions are not supported.
- `RestartPosition` is not supported when the command text contains parameters.
- IBMDADB2 does not quote table names passed through the DBID parameters, which are parameters used by the `IOpenRowset` interface. Instead, the OLE DB consumer must add quotation marks to the table names when quotes are required.

IBM OLE DB Provider support for OLE DB components and interfaces

The following tables list the OLE DB components and interfaces that are supported by the IBM OLE DB Provider for DB2 and the Microsoft OLE DB Provider for ODBC.

Table 13. Blobs

Interface	DB2	ODBC Provider
ISequentialStream	Yes	Yes

Table 14. Commands

Interface	DB2	ODBC Provider
IAccessor	Yes	Yes
ICommand	Yes	Yes
ICommandPersist	No	No
ICommandPrepare	Yes	Yes
ICommandProperties	Yes	Yes
ICommandText	Yes	Yes
ICommandWithParameters	Yes	Yes
IColumnsInfo	Yes	Yes
IColumnsRowset	Yes	Yes
IConvertType	Yes	Yes
ISupportErrorInfo	Yes	Yes

Table 15. DataSources

Interface	DB2	ODBC Provider
IConnectionPoint	No	Yes
IDBAsynchNotify (consumer)	No	No
IDBAsynchStatus	No	No
IDBConnectionPointContainer	No	Yes
IDBCreateSession	Yes	Yes
IDBDataSourceAdmin	No	No
IDBInfo	Yes	Yes
IDBInitialize	Yes	Yes
IDBProperties	Yes	Yes
IPersist	Yes	No
IPersistFile	Yes	Yes
ISupportErrorInfo	Yes	Yes

Table 16. Enumerators

Interface	DB2	ODBC Provider
IDBInitialize	Yes	Yes
IDBProperties	Yes	Yes
IParseDisplayName	Yes	No

Table 16. Enumerators (continued)

Interface	DB2	ODBC Provider
ISourcesRowset	Yes	Yes
ISupportErrorInfo	Yes	Yes

Table 17. Error Lookup Service

Interface	DB2	ODBC Provider
IErrorLookUp	Yes	Yes

Table 18. Error Objects

Interface	DB2	ODBC Provider
IErrorInfo	Yes	No
ISQLErrorInfo (custom)	Yes	No

Table 19. Multiple Results

Interface	DB2	ODBC Provider
IMultipleResults	Yes	Yes
ISupportErrorInfo	Yes	Yes

Table 20. Rowsets

Interface	DB2	ODBC Provider
IAccessor	Yes	Yes
IColumnsRowset	Yes	Yes
IColumnsInfo	Yes	Yes
IConvertType	Yes	Yes
IChapteredRowset	No	No
IConnectionPointContainer	Yes	Yes
IDBAsynchStatus	No	No
IParentRowset	No	No
IRowset	Yes	Yes
IRowsetChange	Yes	Yes
IRowsetChapterMember	No	No
IRowsetFind	No	No
IRowsetIdentity	Yes	Yes
IRowsetIndex	No	No
IRowsetInfo	Yes	Yes
IRowsetLocate	Yes	Yes
IRowsetNotify (consumer)	Yes	No
IRowsetRefresh	Cursor Service Component	Yes
IRowsetResynch	Cursor Service Component	Yes
IRowsetScroll	Yes[1]	Yes
IRowsetUpdate	Cursor Service Component	Yes

Table 20. Rowsets (continued)

Interface	DB2	ODBC Provider
IRowsetView	No	No
ISupportErrorInfo	Yes	Yes
Note: 1. The values to be returned are approximations. Deleted rows will not be skipped.		

Table 21. Sessions

Interface	DB2	ODBC Provider
IAlterIndex	No	No
IAlterTable	No	No
IDBCreateCommand	Yes	Yes
IDBSchemaRowset	Yes	Yes
IGetDataSource	Yes	Yes
IIndexDefinition	No	No
IOpenRowset	Yes	Yes
ISessionProperties	Yes	Yes
ISupportErrorInfo	Yes	Yes
ITableDefinition	No	No
ITableDefinitionWithConstraints	No	No
ITransaction	Yes	Yes
ITransactionJoin	Yes	Yes
ITransactionLocal	Yes	Yes
ITransactionObject	No	No
ITransactionOptions	No	Yes

Table 22. View Objects

Interface	DB2	ODBC Provider
IViewChapter	No	No
IViewFilter	No	No
IViewRowset	No	No
IViewSort	No	No

IBM OLE DB Provider support for OLE DB properties

The following table shows the OLE DB properties that are supported by the IBM OLE DB Provider for DB2:

Table 23. Properties Supported by the IBM OLE DB Provider for DB2: Data Source (DBPROPSET_DATASOURCE)

Properties	Default Value	R/W
DBPROP_MULTIPLECONNECTIONS	VARIANT_FALSE	R
DBPROP_RESETDATASOURCE	DBPROPVAL_RD_RESETALL	R/W

Table 24. Properties Supported by the IBM OLE DB Provider for DB2: DB2 Data Source (DBPROPSET_DB2DATASOURCE)

Properties	Default Value	R/W
DB2PROP_REPORTISLONGFORLONGTYPES	VARIANT_FALSE	R/W
DB2PROP_RETURNCHARASWCHAR	VARIANT_TRUE	R/W
DB2PROP_SORTBYORDINAL	VARIANT_FALSE	R/W

Table 25. Properties Supported by the IBM OLE DB Provider for DB2: Data Source Information (DBPROPSET_DATASOURCEINFO)

Properties	Default Value	R/W
DBPROP_ACTIVESESSIONS	0	R
DBPROP_ASYNCTXNABORT	VARIANT_FALSE	R
DBPROP_ASYNCTXNCOMMIT	VARIANT_FALSE	R
DBPROP_BYREFACCESSORS	VARIANT_FALSE	R
DBPROP_COLUMNDEFINITION	DBPROPVAL_CD_NOTNULL	R
DBPROP_CONCATNULLBEHAVIOR	DBPROPVAL_CB_NULL	R
DBPROP_CONNECTIONSTATUS	DBPROPVAL_CS_INITIALIZED	R
DBPROP_DATASOURCENAME	N/A	R
DBPROP_DATASOURCEREADONLY	VARIANT_FALSE	R
DBPROP_DBMSNAME	N/A	R
DBPROP_DBMSVER	N/A	R
DBPROP_DSOTHREADMODEL	DBPROPVAL_RT_FREETHREAD	R
DBPROP_GROUPBY	DBPROPVAL_GB_CONTAINS_SELECT	R
DBPROP_IDENTIFIERCASE	DBPROPVAL_IC_UPPER	R
DBPROP_MAXINDEXSIZE	0	R
DBPROP_MAXROWSIZE	0	R
DBPROP_MAXROWSIZEINCLUDESBLOB	VARIANT_TRUE	R
DBPROP_MAXTABLEINSELECT	0	R
DBPROP_MULTIPLEPARAMSETS	VARIANT_FALSE	R
DBPROP_MULTIPLERESULTS	DBPROPVAL_MR_SUPPORTED	R
DBPROP_MULTIPLESTORAGEOBJECTS	VARIANT_TRUE	R
DBPROP_MULTITABLEUPDATE	VARIANT_FALSE	R
DBPROP_NULLCOLLATION	DBPROPVAL_NC_LOW	R
DBPROP_OLEOBJECTS	DBPROPVAL_OO_BLOB	R
DBPROP_ORDERBYCOLUMNSINSELECT	VARIANT_FALSE	R
DBPROP _OUTPUTPARAMETERAVAILABILITY	DBPROPVAL_OA_ATEXECUTE	R
DBPROP_PERSISTENTIDTYPE	DBPROPVAL_PT_NAME	R
DBPROP_PREPAREABORTBEHAVIOR	DBPROPVAL_CB_DELETE	R
DBPROP_PROCEDURETERM	"STORED PROCEDURE"	R
DBPROP_PROVIDERFRIENDLYNAME	"IBM OLE DB Provider for DB2"	R

Table 25. Properties Supported by the IBM OLE DB Provider for DB2: Data Source Information (DBPROPSET_DATASOURCEINFO) (continued)

Properties	Default Value	R/W
DBPROP_PROVIDERNAME	"IBMDADB2.DLL"	R
DBPROP_PROVIDEROLEDBVER	"02.7"	R
DBPROP_PROVIDERVER	N/A	R
DBPROP_QUOTEIDENTIFIERCASE	DBPROPVAL_IC_SENSITIVE	R
DBPROP _ROWSETCONVERSIONSONCOMMAND	VARIANT_TRUE	R
DBPROP_SCHEMATERM	"SCHEMA"	R
DBPROP_SCHEMAUSAGE	DBPROPVAL_SU_DML_STATEMENTS | DBPROPVAL_SU_TABLE_DEFINITION | DBPROPVAL_SU_INDEX_DEFINITION | DBPROPVAL_SU_PRIVILEGE_DEFINITION	R
DBPROP_SQLSUPPORT	DBPROPVAL_SQL_ODBC_EXTENDED | DBPROPVAL_SQL_ESCAPECLAUSES | DBPROPVAL_SQL_ANSI92_ENTRY	R
DBPROP_SERVERNAME	N/A	R
DBPROP_STRUCTUREDSTORAGE	DBPROPVAL_SS_ISEQUENTIALSTREAM	R
DBPROP_SUBQUERIES	DBPROPVAL_SQ_CORRELATEDSUBQUERIES | DBPROPVAL_SQ_COMPARISON | DBPROPVAL_SQ_EXISTS | DBPROPVAL_SQ_IN | DBPROPVAL_SQ_QUANTIFIED |	R
DBPROP_SUPPORTEDTXNDDL	DBPROPVAL_TC_ALL	R
DBPROP_SUPPORTEDTXNISOLEVELS	DBPROPVAL_TI_CURSORSTABILITY | DBPROPVAL_TI_READCOMMITTED | DBPROPVAL_TI_READUNCOMMITTED | DBPROPVAL_TI_SERIALIZABLE |	R
DBPROP_SUPPORTEDTXNISORETAIN	DBPROPVAL_TR_COMMIT_DC | DBPROPVAL_TR_ABORT_NO |	R
DBPROP_TABLETERM	"TABLE"	R
DBPROP_USERNAME	N/A	R

Table 26. Properties Supported by the IBM OLE DB Provider for DB2: Initialization (DBPROPSET_DBINIT)

Properties	Default Value	R/W
DBPROP_AUTH_PASSWORD	N/A	R/W
DBPROP_INIT_TIMEOUT (1)	0	R/W
DBPROP_AUTH_PERSIST _SENSITIVE_AUTHINFO	VARIANT_FALSE	R/W
DBPROP_AUTH_USERID	N/A	R/W
DBPROP_INIT_DATASOURCE	N/A	R/W

Table 26. Properties Supported by the IBM OLE DB Provider for DB2: Initialization (DBPROPSET_DBINIT) (continued)

Properties	Default Value	R/W
DBPROP_INIT_HWND	N/A	R/W
DBPROP_INIT_MODE	DB_MODE_READWRITE	R/W
DBPROP_INIT_OLEDBSERVICES	0xFFFFFFFF	R/W
DBPROP_INIT_PROMPT	DBPROMPT_NOPROMPT	R/W
DBPROP_INIT_PROVIDERSTRING	N/A	R/W

Table 27. Properties Supported by the IBM OLE DB Provider for DB2: Rowset (DBPROPSET_ROWSET)

Properties	Default Value	R/W
DBPROP_ABORTPRESERVE	VARIANT_FALSE	R
DBPROP_ACCESSORDER	DBPROPVAL_AO_RANDOM	R
DBPROP_BLOCKINGSTORAGEOBJECTS	VARIANT_FALSE	R
DBPROP_BOOKMARKS	VARIANT_FALSE	R/W
DBPROP_BOOKMARKSKIPPED	VARIANT_FALSE	R
DBPROP_BOOKMARKTYPE	DBPROPVAL_BMK_NUMERIC	R
DBPROP_CACHEDEFERRED	VARIANT_FALSE	R/W
DBPROP_CANFETCHBACKWARDS	VARIANT_FALSE	R/W
DBPROP_CANHOLDROWS	VARIANT_FALSE	R
DBPROP_CANSCROLLBACKWARDS	VARIANT_FALSE	R/W
DBPROP_CHANGEINSERTEDROWS	VARIANT_FALSE	R
DBPROP_COMMITPRESERVE	VARIANT_TRUE	R/W
DBPROP_COMMANDTIMEOUT	0	R/W
DBPROP_DEFERRED	VARIANT_FALSE	R
DBPROP_IAccessor	VARIANT_TRUE	R
DBPROP_IColumnsInfo	VARIANT_TRUE	R
DBPROP_IColumnsRowset	VARIANT_TRUE	R/W
DBPROP_IConvertType	VARIANT_TRUE	R
DBPROP_IMultipleResults	VARIANT_FALSE	R/W
DBPROP_IRowset	VARIANT_TRUE	R
DBPROP_IRowChange	VARIANT_FALSE	R/W
DBPROP_IRowsetFind	VARIANT_FALSE	R
DBPROP_IRowsetIdentity	VARIANT_TRUE	R
DBPROP_IRowsetInfo	VARIANT_TRUE	R
DBPROP_IRowsetLocate	VARIANT_FALSE	R/W
DBPROP_IRowsetScroll	VARIANT_FALSE	R/W
DBPROP_IRowsetUpdate	VARIANT_FALSE	R
DBPROP_ISequentialStream	VARIANT_TRUE	R
DBPROP_ISupportErrorInfo	VARIANT_TRUE	R
DBPROP_LITERALBOOKMARKS	VARIANT_FALSE	R
DBPROP_LITERALIDENTITY	VARIANT_TRUE	R

Table 27. Properties Supported by the IBM OLE DB Provider for DB2: Rowset (DBPROPSET_ROWSET) (continued)

Properties	Default Value	R/W
DBPROP_LOCKMODE	DBPROPVAL_LM_SINGLEROW	R/W
DBPROP_MAXOPENROWS	32767	R
DBPROP_MAXROWS	0	R/W
DBPROP_NOTIFICATIONGRANULARITY	DBPROPVAL_NT_SINGLEROW	R/W
DBPROP_NOTIFICATION PHASES	DBPROPVAL_NP_OKTODO DBPROPBAL_NP_ABOUTTODO DBPROPVAL_NP_SYNCHAFTER DBPROPVAL_NP_FAILEDTODO DBPROPVAL_NP_DIDEVENT	R
DBPROP_NOTIFYROWSETRELEASE	DBPROPVAL_NP_OKTODO DBPROPVAL_NP_ABOUTTODO	R
DBPROP _NOTIFYROWSETFETCHPOSITIONCHANGE	DBPROPVAL_NP_OKTODO DBPROPVAL_NP_ABOUTTODO	R
DBPROP_NOTIFYCOLUMNSET	DBPROPVAL_NP_OKTODO DBPROPVAL_NP_ABOUTTODO	R
DBPROP_NOTIFYROWDELETE	DBPROPVAL_NP_OKTODO DBPROPVAL_NP_ABOUTTODO	R
DBPROP_NOTIFYROWINSERT	DBPROPVAL_NP_OKTODO DBPROPVAL_NP_ABOUTTODO	R
DBPROP_ORDEREDBOOKMARKS	VARIANT_FALSE	R
DBPROP_OTHERINSERT	VARIANT_FALSE	R
DBPROP_OTHERUPDATEDELETE	VARIANT_FALSE	R/W
DBPROP_OWNINSERT	VARIANT_FALSE	R
DBPROP_OWNUPDATEDELETE	VARIANT_FALSE	R
DBPROP_QUICKRESTART	VARIANT_FALSE	R/W
DBPROP_REMOVEDELETED	VARIANT_FALSE	R/W
DBPROP_ROWTHREADMODEL	DBPROPVAL_RT_FREETHREAD	R
DBPROP_SERVERCURSOR	VARIANT_TRUE	R
DBPROP_SERVERDATAONINSERT	VARIANT_FALSE	R
DBPROP_UNIQUEROWS	VARIANT_FALSE	R/W
DBPROP_UPDATABILITY	0	R/W

Table 28. Properties Supported by the IBM OLE DB Provider for DB2: DB2 Rowset (DBPROPSET_DB2ROWSET)

Properties	Default Value	R/W
DBPROP_ISLONGMINLENGTH	32000	R/W

Table 29. Properties Supported by the IBM OLE DB Provider for DB2: Session (DBPROPSET_SESSION)

Properties	Default Value	R/W
DBPROP_SESS_AUTOCOMMITISOLEVELS	DBPROPVAL_TI_CURSORSTABILITY	R/W

Note:

1. The timeout is applicable only when using the TCP/IP protocol to connect to the server. The timeout is enforced only during the TCP/IP sock connect. If the sock connect completes before the specified timeout expires, the timeout will no longer be enforced for the rest of the initialization process. If the client-reroute feature is used then the timeout will be doubled. In general, when client reroute is enabled, the connection timeout behavior is dictated by client reroute.

Connections to data sources using the IBM OLE DB Provider

The following examples show how to connect to a DB2 data source using the IBM OLE DB Provider for DB2.

Example 1: Visual Basic application using ADO

```
Dim db As ADODB.Connection
Set db = New ADODB.Connection
db.Provider = "IBMDADB2"
db.CursorLocation = adUseClient
...
```

Example 2: C/C++ application using IDataInitialize and Service Component

```
hr = CoCreateInstance (
     CLSID_MSDAINITIALIZE,
     NULL,
     CLSCTX_INPROC_SERVER,
     IID_IDataInitialize,
     (void**)&pIDataInitialize);

hr = pIDataInitialize->CreateDBInstance(
     CLSID_IBMDADB2, // ClassID of IBMDADB2
     NULL,
     CLSCTX_INPROC_SERVER,
     NULL,
     IID_IDBInitialize,
     (IUnknown**)&pIDBInitialize);
```

ADO applications

ADO connection string keywords

To specify ActiveX Data Objects (ADO) connection string keywords, you must specify the keyword by using the `keyword=value` format in the provider (connection) string. You must delimit multiple keywords with a semicolon (;).

The following table describes the keywords supported by the IBM OLE DB Provider for DB2:

Table 30. Keywords supported by the IBM OLE DB Provider for DB2

Keyword	Value	Meaning
DSN	Name of the database alias	The DB2 database alias in the database directory.
UID	User ID	The user ID used to connect to the DB2 server.
PWD	Password of UID	Password for the user ID used to connect to the DB2 server.

Other CLI configuration keywords also affect the behavior of the IBM OLE DB Provider.

Connections to data sources with Visual Basic ADO applications

You must specify the IBMDADB2 provider name to connect to a DB2 data source by using the IBM OLE DB Provider for DB2 databases.

Updatable scrollable cursors in ADO applications

If you want to write an ADO application that accesses updatable scrollable cursors, you can set the cursor location to either `adUseClient` or `adUseServer`. If you set the cursor location to `adUseServer`, the cursor materializes on the server.

The IBM OLE DB Provider for DB2 natively supports read-only, forward-only, read-only scrollable, and updatable scrollable cursors.

Limitations for ADO applications

ADO applications have limitations with calling stored procedures, inserting a new row by using a server-side scrollable cursor, and no support for default parameter values.

The limitations for ADO applications are:

- ADO applications calling stored procedures must have their parameters created and explicitly bound. The `Parameters.Refresh` method for automatically generating parameters is not supported for DB2 Server for VSE & VM.
- There is no support for default parameter values.
- When inserting a new row using a server-side scrollable cursor, use the AddNew() method with the Fieldlist and Values arguments. This is more efficient than calling AddNew() with no arguments following Update() calls for each column. Each AddNew() and Update() call is a separate request to the server and therefore, is less efficient than a single call to AddNew().
- Newly inserted rows are not updatable with a server-side scrollable cursor.
- Tables with long or LOB data are not updatable when using a server-side scrollable cursor.

IBM OLE DB Provider support for ADO methods and properties

The IBM OLE DB Provider supports the following ADO methods and properties:

Table 31. Command Methods

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
Cancel	ICommand	Yes
CreateParameter		Yes
Execute		Yes

Table 32. Command Properties

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
ActiveConnection	(ADO specific)	
Command Text	ICommandText	Yes
Command Timeout	ICommandProperties::SetProperties DBPROP_COMMANDTIMEOUT	Yes
CommandType	(ADO specific)	
Prepared	ICommandPrepare	Yes
State	(ADO specific)	

Table 33. Command Collections

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
Parameters	ICommandWithParameter DBSCHEMA _PROCEDURE_PARAMETERS	Yes
Properties	ICommandProperties IDBProperties	Yes

Table 34. Connection Methods

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
BeginTrans CommitTrans RollbackTrans	ITransactionLocal	Yes (but not nested) Yes (but not nested) Yes (but not nested)
Execute	ICommand IOpenRowset	Yes
Open	IDBCreateSession IDBInitialize	Yes

Table 34. Connection Methods (continued)

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
OpenSchema adSchemaColumnPrivileges adSchemaColumns adSchemaForeignKeys adSchemaIndexes adSchemaPrimaryKeys adSchemaProcedureParam adSchemaProcedures adSchemaProviderType adSchemaStatistics adSchemaTablePrivileges adSchemaTables	IDBSchemaRowset	Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
Cancel		Yes

Table 35. Connection Properties

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
Attributes adXactCommitRetaining adXactRollbackRetaining	ITransactionLocal	Yes Yes
CommandTimeout	ICommandProperties DBPROP_COMMAND_TIMEOUT	Yes
ConnectionString	(ADO specific)	
ConnectionTimeout	IDBProperties DBPROP_INIT_TIMEOUT	No
CursorLocation: adUseClient adUseNone adUseServer	(Use OLE DB Cursor Service) (Not Used)	Yes No Yes
DefaultDataBase	IDBProperties DBPROP_CURRENTCATALOG	No
IsolationLevel	ITransactionLocal DBPROP_SESS _AUTOCOMMITISOLEVELS	Yes
Mode adModeRead adModeReadWrite adModeShareDenyNone adModeShareDenyRead adModeShareDenyWrite adModeShareExclusive adModeUnknown adModeWrite	IDBProperties DBPROP_INIT_MODE	No Yes No No No No No No
Provider	ISourceRowset::GetSourceRowset	Yes

Table 35. Connection Properties (continued)

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
State	(ADO specific)	
Version	(ADO specific)	

Table 36. Connection Collection

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
Errors	IErrorRecords	Yes
Properties	IDBProperties	Yes

Table 37. Error Properties

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
Description NativeError Number Source SQLState	IErrorRecords	Yes Yes Yes Yes Yes
HelpContext HelpFile		No No

Table 38. Field Methods

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
AppendChunk GetChunk	ISequentialStream	Yes Yes

Table 39. Field Properties

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
Actual Size	IAccessor IRowset	Yes
Attributes DataFormat DefinedSize Name NumericScale Precision Type	IColumnInfo	Yes Yes Yes Yes Yes Yes
OriginalValue	IRowsetUpdate	Yes (Cursor Service)
UnderlyingValue	IRowsetRefresh IRowsetResynch	Yes (Cursor Service) Yes (Cursor Service)

Table 39. Field Properties (continued)

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
Value	IAccessor IRowset	Yes

Table 40. Field Collection

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
Properties	IDBProperties IRowsetInfo	Yes

Table 41. Parameter Methods

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
AppendChunk	ISequentialStream	Yes
Attributes Direction Name NumericScale Precision Scale Size Type	ICommandWithParameter DBSCHEMA _PROCEDURE_PARAMETERS	Yes No Yes Yes Yes Yes Yes
Value	IAccessor ICommand	Yes

Table 42. Parameter Collection

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
Properties		Yes

Table 43. RecordSet Methods

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
AddNew	IRowsetChange	Yes
Cancel		Yes
CancelBatch	IRowsetUpdate::Undo	Yes (Cursor Service)
CancelUpdate		Yes (Cursor Service)
Clone	IRowsetLocate	Yes
Close	IAccessor IRowset	Yes
CompareBookmarks		No
Delete	IRowsetChange	Yes
GetRows	IAccessor IRowset	Yes

Table 43. RecordSet Methods (continued)

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
Move	IRowset IRowsetLocate	Yes
MoveFirst	IRowset IRowsetLocate	Yes
MoveNext	IRowset IRowsetLocate	Yes
MoveLast	IRowsetLocate	Yes
MovePrevious	IRowsetLocate	Yes
NextRecordSet	IMultipleResults	Yes
Open	ICommand IOpenRowset	Yes
Requery	ICommand IOpenRowset	Yes
Resync	IRowsetRefresh	Yes (Cursor Service)
Supports	IRowsetInfo	Yes
Update UpdateBatch	IRowsetChange IRowsetUpdate	Yes Yes (Cursor Service)

Table 44. RecordSet Properties

Method/Property	OLE DB Interface/Property	IBM OLE DB Support
AbsolutePage	IRowsetLocate IRowsetScroll	Yes Yes[1]
AbsolutePosition	IRowsetLocate IRowsetScroll	Yes Yes[1]
ActiveConnection	IDBCreateSession IDBInitialize	Yes
BOF	(ADO specific)	
Bookmark	IAccessor IRowsetLocate	Yes
CacheSize	cRows in IRowsetLocate IRowset	Yes
CursorType adOpenDynamic adOpenForwardOnly adOpenKeySet adOpenStatic	ICommandProperties	No Yes Yes Yes

Table 44. RecordSet Properties (continued)

Method/Property	**OLE DB Interface/Property**	**IBM OLE DB Support**
EditMode	IRowsetUpdate	Yes (Cursor Service)
EOF	(ADO specific)	
Filter	IRowsetLocate IRowsetView IRowsetUpdate IViewChapter IViewFilter	No
LockType	ICommandProperties	Yes
MarshallOption		No
MaxRecords	ICommandProperties IOpenRowset	Yes
PageCount	IRowsetScroll	Yes[1]
PageSize	(ADO specific)	
Sort	(ADO specific)	
Source	(ADO specific)	
State	(ADO specific)	
Status	IRowsetUpdate	Yes (Cursor Service)

Note:

1. The values to be returned are approximations. Deleted rows will not be skipped.

Table 45. RecordSet Collection

Method/Property	**OLE DB Interface/Property**	**IBM OLE DB Support**
Fields	IColumnInfo	Yes
Properties	IDBProperties IRowsetInfo::GetProperties	Yes

Compilation and linking of C/C++ applications and the IBM OLE DB Provider

If you write C/C++ applications that use the constant CLSID_IBMDADB2, you must include the `ibmdadb2.h` header file. You can find the header file in the `SQLLIB\include` directory.

These applications must define `DBINITCONSTANTS` before the include statement. The following example shows the correct sequence of C/C++ preprocessor directives:

```
#define DBINITCONSTANTS
#include "ibmdadb2.h"
```

Connections to data sources in C/C++ applications using the IBM OLE DB Provider

To connect to a DB2 data source with the IBM OLE DB provider (IBMDADB2) in a C/C++ application, you can use the IDBPromptInitialize or IDataInitialize interface. If you do not want to use the interfaces, you can call the CoCreateInstance API (COM).

The IDataInitialize interface is exposed by the OLE DB Service Component, and the IDBPromptInitialize is exposed by the Data Links Component.

COM+ distributed transaction support and the IBM OLE DB Provider

OLE DB applications running in a Microsoft Component Services (COM+) environment on Windows 2000 can use the `ITransactionJoin` interface to participate in distributed transactions with multiple DB2 for Linux, UNIX, and Windows, host, and System i® database servers as well as other resource managers that comply with the COM+ specifications.

Prerequisites

To use the COM+ distributed transaction support offered by the IBM OLE DB Provider for DB2, ensure that your server meets the following prerequisites.

Note: These requirements are only for the Windows-based computers where DB2 clients are installed.

- Windows 2000 with Service Pack 3 or later

Enablement of COM+ support in C/C++ database applications

To run a C or C++ application in COM+ transactional mode, you can create the IBMDADB2 data source instance using `CoCreateInstance`, get a session object, and use `JoinTransaction`. See the description of how to connect a C or C++ application to a data source for more information.

To run an ADO application in COM+ transactional mode, see the description of how to connect a C or C++ application to a data source.

To use a component in an COM+ package in transactional mode, set the Transactions property of the component to one of the following values:

- "Required"
- "Required New"
- "Supported"

For information about these values, see the COM+ documentation.

Chapter 3. IBM Data Server Provider for .NET namespaces

The IBM Data Server Provider for .NET extends support for the Microsoft .NET Framework with the use of the IBM namespaces. The namespace documentation includes a summary of all classes and associated members.

Namespace	Description
IBM.Data.DB2	Provides classes that are required to access DB2 and Informix database servers.
IBM.Data.DB2Types	Provides classes for native data types in IBM database servers.
IBM.Data.Informix	Provides classes that are required to access Informix database servers.

Important: The Informix .NET provider (IBM.Data.Informix.dll) is deprecated since DB2 Version 10.1 Fix Pack 1 and might be discontinued in a later release. Start using the DB2 .NET provider (IBM.Data.DB2.dll) to connect to Informix database servers.

IBM.Data.DB2 namespace

The IBM.Data.DB2 namespace contains classes that are associated with the DB2 .NET provider. You can use the classes for connecting to a database, executing commands, and retrieving results.

To use the DB2 .NET provider, you must add an `imports` or `using` statement for the IBM.Data.DB2 namespace to your application:

```
[Visual Basic]
Imports IBM.Data.DB2
[C#]
using IBM.Data.DB2;
```

You also must include a reference to the `IBM.Data.DB2.dll` file in your application's project. In Visual Studio, you can include this reference by using the References section for your project in Solution Explorer or by clicking **Project** > **Add Reference**. If you are compiling a program from the command line, you can specify the IBM .NET provider file by specifying the following option for the **csc** or **vbc** command:

```
/r:install_dir\bin\netfXX\IBM.Data.IBM.Data.DB2.dll
```

To establish a connection to one of the supported data servers, you must construct a DB2Connection object and provide it with a valid DB2 .NET connection string. For information about the supported keywords, see DB2Connection.ConnectionString topic.

You require the following classes to access the data in database servers:
- DB2DataAdapter
- DB2Command
- DB2Connection
- DB2DataReader

Classes

Class	Description
DB2BulkCopy	Facilitates the copying of rows from one data source to another.
DB2BulkCopyColumnMapping	Represents a column mapping from the data source table to the destination table.
DB2BulkCopyColumnMappingCollection	Represents a collection of column mappings from the data source table to the destination table.
DB2Command	Represents an SQL statement or stored procedure to execute against a database.
DB2CommandBuilder	Automatically generates single-table commands that are used to reconcile changes that are made to the DataSet object.
DB2Connection	Represents an open connection to a database.
DB2ConnectionStringBuilder	Facilitates generic and IBM Data Server Provider for .NET-specific approaches to generating valid connection strings.
DB2DataAdapter	Represents a set of data commands and a connection to a database that are used to fill the DataSet object and update the database.
DB2DataReader	Provides a way of reading a forward-only stream of data rows from a database.
DB2DataSourceEnumerator	Provides a way to discover available IBM family data sources.
DB2Error	Collects a database warning or a database error.
DB2ErrorCollection	Collects all errors that are generated by the DB2DataAdapter object.
DB2Exception	Represents exception that is generated when an error is returned by an IBM database server.
DB2Factory	Represents a set of methods for creating instances of the System.Data.Common data source classes for the data provider.
DB2InfoMessageEventArgs	Provides data for the InfoMessage event.
DB2Parameter	Represents a parameter to a DB2Command object and, optionally, its mapping to a DataColumn object.
DB2ParameterCollection	Represents a collection of parameters that are relevant to a DB2Command object and their mappings to columns in a DataSet object.
DB2Permission	Enables the IBM Data Server Provider for .NET to ensure that a user has a security level that is adequate to access an IBM database.
DB2PermissionAttribute	Associates a security action with a custom security attribute.
DB2Record	Represents a read-only record.
DB2ResultSet	Provides the ability to scroll through a bindable stream of rows that are returned from a database. You can also insert, update, and delete rows in the DB2ResultSet object.

Class	Description
DB2RowsCopiedEventArgs	Provides data for the DB2RowsCopied event.
DB2RowUpdatedEventArgs	Provides data for the RowUpdated event.
DB2RowUpdatingEventArgs	Provides data for the RowUpdating event.
DB2Transaction	Represents an SQL transaction.
DB2UpdatableRecord	Represents a row to be created in a DB2ResultSet object.
DB2XmlAdapter	Populates the XPathDocument object with data that is retrieved from a database.

Delegates

Delegate	Description
DB2InfoMessageEventHandler	Represents the method that handles the InfoMessage event from the DB2Connection object.
DB2RowsCopiedEventHandler	Represents the method that handles the DB2RowsCopied event from the DB2BulkCopy object.
DB2RowUpdatedEventHandler	Represents the method that handles the RowUpdated event from the DB2DataAdapter object.
DB2RowUpdatingEventHandler	Represents the method that handles the RowUpdating event from the DB2DataAdapter object.

Enumerations

Enumeration	Description
DB2BulkCopyOptions	Specifies options to use with the DB2BulkCopy object. The DB2BulkCopyOptions enumeration consists of bit flags, which you can combine with a bitwise operation.
DB2CursorType	Specifies cursor options to use with the DB2ResultSet object. The DB2CursorType enumeration consists of bit flags, which you can combine with a bitwise operation.
DB2ResultSetOptions	Specifies result set options to use with the DB2ResultSet object. The DB2ResultSetOptions enumeration consists of bit flags, which you can combine with a bitwise operation.
DB2Type	Specifies the data type of a field, property, or DB2Parameter object.

Reference

"IBM.Data.DB2Types Namespace" on page 78
The IBM.Data.DB2Types namespace provides classes and structures that represent DB2 data types for the .NET Framework Data Provider.

Chapter 3, “IBM Data Server Provider for .NET namespaces,” on page 75
The IBM Data Server Provider for .NET extends support for the Microsoft .NET Framework with the use of the IBM namespaces. The namespace documentation includes a summary of all classes and associated members.

IBM.Data.DB2Types Namespace

The IBM.Data.DB2Types namespace provides classes and structures that represent DB2 data types for the .NET Framework Data Provider.

The following table shows mappings between DB2Type data types, DB2 data types, Informix data types, Microsoft .NET Framework types, and DB2Types classes and structures.

Table 46. Mappings between data types, classes and structures

Category	DB2Types Classes and Structures	DB2Type Data Type	DB2 Data Type	Informix Data Type	.NET Data Type
Numeric	DB2Int16	SmallInt	SMALLINT	BOOLEAN, SMALLINT	Int16
	DB2Int32	Integer	INT	INTEGER, INT, SERIAL	Int32
	DB2Int64	BigInt, BigSerial	BIGINT	BIGINT, BIGSERIAL, INT8, SERIAL8	Int64
	DB2Real, DB2Real370	Real	REAL	REAL, SMALLFLOAT	Single
	DB2Double	Double	DOUBLE PRECISION	DECIMAL (≤31), DOUBLE PRECISION	Double
	DB2Double	Float	FLOAT	DECIMAL (32), FLOAT	Double
	DB2Decimal	Decimal	DECIMAL	MONEY	Decimal
	DB2Decimal Float	DecimalFloat	DECFLOAT (16|34)		Decimal
	DB2Decimal	Numeric	DECIMAL	DECIMAL (≤31), NUMERIC	Decimal
Date/Time	DB2Date	Date	DATE	DATETIME (date precision)	Datetime
	DB2Time	Time	TIME	DATETIME (time precision)	TimeSpan
	DB2 TimeStamp	Timestamp	TIMESTAMP	DATETIME (time and date precision)	DateTime
	DB2 TimeStampOffset	Timestamp WithTimeZone	TIMESTAMP WITH TIME ZONE	N/A	DateTime Offset
XML	DB2Xml	XmlIfxType.Xml	XML		Byte[]
Character data	DB2String	Char	CHAR	CHAR	String
	DB2String	VarChar	VARCHAR	VARCHAR	String
	DB2String	LongVarChar	LONG VARCHAR	LVARCHAR	String
Binary data	DB2Binary	Binary	CHAR FOR BIT DATA		Byte[]
	DB2Binary	Binary	BINARY		Byte[]
	DB2Binary	VarBinary	VARBINARY		Byte[]
	DB2Binary	Long VarBinary	LONG VARCHAR FOR BIT DATA		Byte[]
Graphic data	DB2String	Graphic	GRAPHIC		String
	DB2String	VarGraphic	VARGRAPHIC		String
	DB2String	Long VarGraphic	LONG VARGRAPHIC		String

6. These data types are not supported as parameters in DB2 .NET common language runtime routines.
7. A DB2ParameterClass.ParameterName property of the type DB2Type.Xml can accept variables of the following types: String, byte[], DB2Xml, and XmlReader.
8. These data types are applicable only to DB2 for z/OS Version 9 and later releases and DB2 for i V6R1 and later releases.
9. This data type is only supported for DB2 for z/OS Version 9 and later releases and for DB2 for Linux, UNIX, and Windows Version 9.5 and later releases.

Table 46. Mappings between data types, classes and structures (continued)

Category	DB2Types Classes and Structures	DB2Type Data Type	DB2 Data Type	Informix Data Type	.NET Data Type
LOB data	DB2Clob	Clob	CLOB	CLOB, TEXT	String
	DB2Blob	Blob	BLOB	BLOB, BYTE	Byte[]
	DB2Clob	DbClob	DBCLOB		String
Row ID	DB2RowId	RowId	ROWID		Byte[]

Enumerations

Enumeration	Description
DB2RoundingMode	Specify the rounding mode to use with DB2DecimalFloat objects.

Appendix A. DB2 technical information

DB2 technical information is available in multiple formats that can be accessed in multiple ways.

DB2 technical information is available through the following tools and methods:
- Online DB2 documentation in IBM Knowledge Center:
  - Topics (task, concept, and reference topics)
  - Sample programs
  - Tutorials
- Locally installed DB2 Information Center:
  - Topics (task, concept, and reference topics)
  - Sample programs
  - Tutorials
- DB2 books:
  - PDF files (downloadable)
  - PDF files (from the DB2 PDF DVD)
  - Printed books
- Command-line help:
  - Command help
  - Message help

Important: The documentation in IBM Knowledge Center and the DB2 Information Center is updated more frequently than either the PDF or the hardcopy books. To get the most current information, install the documentation updates as they become available, or refer to the DB2 documentation in IBM Knowledge Center.

You can access additional DB2 technical information such as technotes, white papers, and IBM Redbooks® publications online at ibm.com. Access the DB2 Information Management software library site at http://www.ibm.com/software/data/sw-library/.

Documentation feedback

The DB2 Information Development team values your feedback on the DB2 documentation. If you have suggestions for how to improve the DB2 documentation, send an email to db2docs@ca.ibm.com. The DB2 Information Development team reads all of your feedback but cannot respond to you directly. Provide specific examples wherever possible to better understand your concerns. If you are providing feedback on a specific topic or help file, include the topic title and URL.

Do not use the db2docs@ca.ibm.com email address to contact DB2 Customer Support. If you have a DB2 technical issue that you cannot resolve by using the documentation, contact your local IBM service center for assistance.

DB2 technical library in hardcopy or PDF format

You can download the DB2 technical library in PDF format or you can order in hardcopy from the IBM Publications Center.

English and translated DB2 Version 10.5 manuals in PDF format can be downloaded from DB2 database product documentation at www.ibm.com/support/docview.wss?rs=71&uid=swg27009474.

The following tables describe the DB2 library available from the IBM Publications Center at http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss. Although the tables identify books that are available in print, the books might not be available in your country or region.

The form number increases each time that a manual is updated. Ensure that you are reading the most recent version of the manuals, as listed in the following tables.

The DB2 documentation online in IBM Knowledge Center is updated more frequently than either the PDF or the hardcopy books.

Table 47. DB2 technical information

Name	Form number	Available in print	Availability date
Administrative API Reference	SC27-5506-00	Yes	28 July 2013
Administrative Routines and Views	SC27-5507-01	No	1 October 2014
Call Level Interface Guide and Reference Volume 1	SC27-5511-01	Yes	1 October 2014
Call Level Interface Guide and Reference Volume 2	SC27-5512-01	No	1 October 2014
Command Reference	SC27-5508-01	No	1 October 2014
Database Administration Concepts and Configuration Reference	SC27-4546-01	Yes	1 October 2014
Data Movement Utilities Guide and Reference	SC27-5528-01	Yes	1 October 2014
Database Monitoring Guide and Reference	SC27-4547-01	Yes	1 October 2014
Data Recovery and High Availability Guide and Reference	SC27-5529-01	No	1 October 2014
Database Security Guide	SC27-5530-01	No	1 October 2014
DB2 Workload Management Guide and Reference	SC27-5520-01	No	1 October 2014
Developing ADO.NET and OLE DB Applications	SC27-4549-01	Yes	1 October 2014
Developing Embedded SQL Applications	SC27-4550-00	Yes	28 July 2013

Table 47. DB2 technical information (continued)

Name	Form number	Available in print	Availability date
Developing Java Applications	SC27-5503-01	No	1 October 2014
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-5504-01	No	1 October 2014
Developing RDF Applications for IBM Data Servers	SC27-5505-00	Yes	28 July 2013
Developing User-defined Routines (SQL and External)	SC27-5501-00	Yes	28 July 2013
Getting Started with Database Application Development	GI13-2084-01	Yes	1 October 2014
Getting Started with DB2 Installation and Administration on Linux and Windows	GI13-2085-01	Yes	1 October 2014
Globalization Guide	SC27-5531-00	No	28 July 2013
Installing DB2 Servers	GC27-5514-01	No	1 October 2014
Installing IBM Data Server Clients	GC27-5515-01	No	1 October 2014
Message Reference Volume 1	SC27-5523-00	No	28 July 2013
Message Reference Volume 2	SC27-5524-00	No	28 July 2013
Net Search Extender Administration and User's Guide	SC27-5526-01	No	1 October 2014
Partitioning and Clustering Guide	SC27-5532-01	No	1 October 2014
pureXML Guide	SC27-5521-00	No	28 July 2013
Spatial Extender User's Guide and Reference	SC27-5525-00	No	28 July 2013
SQL Procedural Languages: Application Enablement and Support	SC27-5502-00	No	28 July 2013
SQL Reference Volume 1	SC27-5509-01	No	1 October 2014
SQL Reference Volume 2	SC27-5510-01	No	1 October 2014
Text Search Guide	SC27-5527-01	Yes	1 October 2014
Troubleshooting and Tuning Database Performance	SC27-4548-01	Yes	1 October 2014
Upgrading to DB2 Version 10.5	SC27-5513-01	Yes	1 October 2014
What's New for DB2 Version 10.5	SC27-5519-01	Yes	1 October 2014
XQuery Reference	SC27-5522-01	No	1 October 2014

Table 48. DB2 Connect technical information

Name	Form number	Available in print	Availability date
Installing and Configuring DB2 Connect Servers	SC27-5517-00	Yes	28 July 2013
DB2 Connect User's Guide	SC27-5518-01	Yes	1 October 2014

Displaying SQL state help from the command line processor

DB2 products return an SQLSTATE value for conditions that can be the result of an SQL statement. SQLSTATE help explains the meanings of SQL states and SQL state class codes.

Procedure

To start SQL state help, open the command line processor and enter:

```
? sqlstate or ? class code
```

where *sqlstate* represents a valid five-digit SQL state and *class code* represents the first two digits of the SQL state.
For example, `? 08003` displays help for the 08003 SQL state, and `? 08` displays help for the 08 class code.

Accessing DB2 documentation online for different DB2 versions

You can access online the documentation for all the versions of DB2 products in IBM Knowledge Center.

About this task

All the DB2 documentation by version is available in IBM Knowledge Center at http://www.ibm.com/support/knowledgecenter/SSEPGG/welcome. However, you can access a specific version by using the associated URL for that version.

Procedure

To access online the DB2 documentation for a specific DB2 version:

- To access the DB2 Version 10.5 documentation, follow this URL: http://www.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.kc.doc/welcome.html.
- To access the DB2 Version 10.1 documentation, follow this URL: http://www.ibm.com/support/knowledgecenter/SSEPGG_10.1.0/com.ibm.db2.luw.kc.doc/welcome.html.
- To access the DB2 Version 9.8 documentation, follow this URL: http://www.ibm.com/support/knowledgecenter/SSEPGG_9.8.0/com.ibm.db2.luw.kc.doc/welcome.html.
- To access the DB2 Version 9.7 documentation, follow this URL: http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.kc.doc/welcome.html.

- To access the DB2 Version 9.5 documentation, follow this URL: http://www.ibm.com/support/knowledgecenter/SSEPGG_9.5.0/com.ibm.db2.luw.kc.doc/welcome.html.

Terms and conditions

Appendix B. Notices

This information was developed for products and services offered in the U.S.A. Information about non-IBM products is based on information available at the time of first publication of this document and is subject to change.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country/region where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements, changes, or both in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to websites not owned by IBM are provided for convenience only and do not in any manner serve as an endorsement of those

platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _*enter the year or years*_. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle, its affiliates, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Intel, Intel logo, Intel Inside, Intel Inside logo, Celeron, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Index

Printed in USA

SC27-4549-01

Spine information:

IBM DB2 10.5 for Linux, UNIX, and Windows

Developing ADO.NET and OLE DB Applications