Informix Product Family
Data Server Provider for .NET
Version 9.7

# *IBM Data Server Provider for .NET Programmer's Guide, Informix Edition*

SC27-3518-02

Informix Product Family
Data Server Provider for .NET
Version 9.7

# *IBM Data Server Provider for .NET Programmer's Guide, Informix Edition*

SC27-3518-02

**Note**
Before using this information and the product it supports, read the information in "Notices" on page B-1.

**Edition**

This edition replaces SC27-3518-01.

# Contents

# Introduction

## In This Introduction

This introduction provides an overview of the information in this publication and describes the conventions that this publication uses.

## About This Publication

This publication contains the information that you need to use the IBM® Data Server Provider for .NET so that you can access and manipulate data in IBM Informix® databases. This publication assumes that you are familiar with the Microsoft .NET specification, object-oriented programming principles, and using IBM Informix servers and databases.

Microsoft provides information about programming with .NET on its Web site. For more information about working with IBM Informix Dynamic Server, see the *Getting Started* publication in your server documentation set.

Windows and .NET are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

### Types of Users

This publication is written for the following users:

- .NET application developers
- Database administrators
- System administrators

This publication is written with the assumption that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts
- Some experience with application development, database server administration, operating-system administration, or network administration
- Some experience with .NET technology

If you have limited experience with relational databases, SQL, or your operating system, refer to the *IBM Informix Dynamic Server Getting Started Guide* for your database server for a list of supplementary titles.

### Software Dependencies

This publication is written with the assumption that you are using IBM Informix or IBM Informix Dynamic Server with J/Foundation, 11.70, as your database server.

## Assumptions About Your Locale

IBM Informix products can support many languages, cultures, and code sets. All the information related to character set, collation, and representation of numeric data, currency, date, and time is brought together in a single environment, called a Global Language Support (GLS) locale.

The examples in this publication are written with the assumption that you are using the default locale, **en_us.8859-1**. This locale supports U.S. English format conventions for date, time, and currency. In addition, this locale supports the ISO 8859-1 code set, which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

If you plan to use nondefault characters in your data or your SQL identifiers, or if you want to conform to the nondefault collation rules of character data, you need to specify the appropriate nondefault locale.

For instructions on how to specify a nondefault locale, additional syntax, and other considerations related to GLS locales, see the *IBM Informix GLS User's Guide*.

## Demonstration Databases

The DB-Access utility includes one or more demonstration databases that you can use to learn and test with. After you add, delete, or change the data and scripts that are in the database, you can re-initialize the database to its original condition.

The demonstration databases are:

- The **stores_demo** database illustrates a relational schema with information about a fictitious wholesale sporting-goods distributor. Many examples in IBM Informix publications are based on the **stores_demo** database.
- The **sales_demo** database provides an example of a simple data-warehousing environment and works in conjunction with the stores_demo database. The scripts for the sales_demo database create new tables and add extra rows to the items and orders tables of stores_demo database.
- The **superstores_demo** database illustrates an object-relational schema. The **superstores_demo** database contains examples of extended data types, type and table inheritance, and user-defined routines.

For information about how to create and populate the demonstration databases, see the *IBM Informix DB-Access User's Guide*. For descriptions of the databases and their contents, see the *IBM Informix Guide to SQL: Reference*.

The scripts that you use to install the demonstration databases reside in the **$INFORMIXDIR/bin** directory on UNIX and in the **%INFORMIXDIR%\bin** directory on Windows.

# What's New in IBM Data Server Provider for .NET

This publication includes information about new features and changes in existing functionality.

The following changes and enhancements are relevant to this publication.

The following table lists the new features for Version 9.7 Fix Pack 4.

*Table 1. What's New in the IBM Data Server Provider for .NET for Version 9.7 Fix Pack 4*

| Overview | Reference |
|---|---|
| **Updated Canonical functions**<br><br>IBM Data Server Provider for .NET supports new canonical functions. | |
| **New testconn utility**<br><br>Starting in Version 9.7 Fix Pack 4, IBM Data Server Provider for .NET supports new testconn utility. | See "Validating IBM Data Server Driver Package (Windows) installation"<br><br>See "Validating using testconn utility" |
| **Framework 4.0 support**<br><br>IBM Data Server Provider for .NET supports Framework 4.0. | |
| **Visual Studio 2010 support**<br><br>IBM Visual Studio Add-Ins support Visual Studio 2010. | |
| **FitHighPrecisionType support**<br><br>IBM Data Server Provider for .NET supports a new keyword FitHighPrecisionType. | See "IfxCommand Properties"<br><br>See the list of Keywords in the "IfxConnection.ConnectionString Property"<br><br>See "IfxConnectionStringBuilder.FitHighPrecisionType Property" |
| **Removal of U2 support**<br><br>Data Server Provider for .NET and IBM Visual Studio Add-Ins no longer support U2 servers. | |

The following table lists the new features for Version 9.7 Fix Pack 3a.

*Table 2. What's New in the IBM Data Server Provider for .NET for Version 9.7 Fix Pack 3a*

| Overview | Reference |
|---|---|
| **Differences between .NET providers**<br><br>Additional information about the differences between the IBM Data Server .NET Provider and the IBM Informix .NET Provider was added to this guide, including information about the protocols used and data types supported by each provider. | See "Differences between .NET Providers". |

*Table 2. What's New in the IBM Data Server Provider for .NET for Version 9.7 Fix Pack 3a (continued)*

| Overview | Reference |
|---|---|
| **New editions and product names**<br><br>IBM Informix Dynamic Server editions were withdrawn and new Informix editions are available. Some products were also renamed. The publications in the Informix library pertain to the following products:<br>• IBM Informix database server, formerly known as IBM Informix Dynamic Server (IDS)<br>• IBM OpenAdmin Tool (OAT) for Informix, formerly known as OpenAdmin Tool for Informix Dynamic Server (IDS)<br>• IBM Informix SQL Warehousing Tool, formerly known as Informix Warehouse Feature | For more information about the Informix product family, go to http://www.ibm.com/software/data/informix/. |

The following table lists the new features for Version 9.7 Fix Pack 2.

*Table 3. What's New in the IBM Data Server Provider for .NET for Version 9.7 Fix Pack 2*

| Overview | Reference |
|---|---|
| **32-bit drivers in 64-bit package**<br><br>32-bit versions of IBM Data Server Provider for .NET are included in the 64-bit driver package. When you install the 64-bit drivers, the 32-bit drivers are also installed, in a separate directory named, sqllib\bin\netf20_32. | See "32-bit and 64-bit support for ADO.NET applications" |
| **Database connection synonym processing bypass support**<br><br>You can use a new db2dsdriver.cfg file keyword or connection string property, SkipSynonymProcessing, to bypass synonym processing when opening a connection. Using the keyword or connection string property when you do not require synonym processing can reduce connection time overhead when you use DB2Connection or DB2ConnectionStringBuilder. | See "SkipSynonymProcessing IBM Data Server Driver configuration keyword" |
| **Query timeout support in the db2dsdriver.cfg.file**<br><br>You can use a new db2dsdriver.cfg file keyword, QueryTimeout, as a centralized control to indicate how long a client should wait for a query to run before timing out. | See "QueryTimeoutInterval IBM Data Server Driver configuration keyword" |

*Table 3. What's New in the IBM Data Server Provider for .NET for Version 9.7 Fix Pack 2 (continued)*

| Overview | Reference |
|---|---|
| **DisableCursorHold**<br><br>By default, the cursors that a .NET application opens are preserved even after the application has committed the transaction. This has an unnecessary cost for applications that do not intend to reuse the cursor after a transaction is committed. To alleviate the cost of such applications, a new connection string attribute, a new db2dsdriver.cfg configuration parameter and a new command property named **DisableCursorHold** is available. | See "DisableCursorHold IBM Data Server Driver configuration keyword"<br><br>See "IfxConnectionStringBuilder.DisableCursorHold Property" |
| **Connection string property support**<br><br>IBM Data Server Provider for .NET supports connection string properties that provide client information. | See "IfxConnectionStringBuilder Properties" |

The following table lists the new features for Version 9.7 Fix Pack 1.

*Table 4. What's New in the IBM Data Server Provider for .NET for Version 9.7 Fix Pack 1*

| Overview | Reference |
|---|---|
| **Informix optimizer directives**<br><br>The IBM Data Server Provider for .NET does not process Informix optimizer directives. Starting in Fix Pack 1, the data provider passes the directives through the client-side parsing to the data server where all directive driven optimizations occur. | See "Optimizer Directives" in the *IBM Informix Dynamic Server Guide to SQL: Syntax*. |
| **IfxType.Money**<br><br>Starting in Fix Pack 1, support for the Informix MONEY data type has been added as valid IfxType enumeration. The Money data type will be treated as a DECIMAL with 2 digits of precision. | See IfxType Enumeration |

The following table lists the new features for Version 9.7.

Version 9.7 includes enhancements that improve the performance of .NET applications that connect to Informix Dynamic Server (IDS) data servers.

*Table 5. What's New in the IBM Data Server Provider for .NET for Version 9.7*

| Overview | Reference |
|---|---|
| **BIGINT and BIGSERIAL data types**<br><br>Previously, the data server provider for .NET supported only INT8 and SERIAL8 data types for 64–bit integers. Support has been added for BIGINT and BIGSERIAL for Informix data servers which provide better performance than the INT8 and SERIAL8 data types.The IBM Data Server Provider for .NET now supports BIGINT and BIGSERIAL data types. | • See IBM.Data.Informix Namespace<br>• See DB2Int64 Structure |
| **ReturnValue parameters**<br><br>IDS stored procedures can return single or multiple result sets. Previously, the data server provider for .NET did not support multiple values from user defined routines (UDRs). The added support for ReturnValue parameters means that the data server provider for .NET is able to retrieve the result set as a single return value. | • See IfxCommand.ResultSetAsReturnValue Property<br>• See IfxConnection.ResultSetAsReturnValue Property |
| **High-availability Data Replication**<br><br>The data server provider for .NET now supports High-availability Data Replication (HDR). | See the *IBM Informix Dynamic Server Administrator's Guide* for more information on High-availability Data Replication. |
| **Workload balancing**<br><br>The data server provider for .NET now supports workload balancing (WLB). When an application tries to connect to a server, the Connection Manager routes the request to the server with the lowest CPU usage. | See the "Manage Cluster Connections with the Connection Manager" in the *IBM Informix Dynamic Server Administrator's Guide*. |

## Example code conventions

Examples of SQL code occur throughout this publication. Except as noted, the code is not specific to any single IBM Informix application development tool.

If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the code in the following example:

```
CONNECT TO stores_demo
...

DELETE FROM customer
   WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

To use this SQL code for a specific product, you must apply the syntax rules for that product. For example, if you are using an SQL API, you must use EXEC SQL at the start of each statement and a semicolon (or other appropriate delimiter) at the end of the statement. If you are using DB–Access, you must delimit multiple statements with semicolons.

**Tip:** Ellipsis points in a code example indicate that more code would be added in a full application, but it is not necessary to show it to describe the concept being discussed.

For detailed directions on using SQL statements for a particular application development tool or SQL API, see the documentation for your product.

## Additional documentation

Documentation about this release of IBM Informix products is available in various formats.

You can access or install the product documentation from the Quick Start CD that is shipped with Informix products. To get the most current information, see the Informix information centers at ibm.com®. You can access the information centers and other Informix technical information such as technotes, white papers, and IBM Redbooks® publications online at http://www.ibm.com/software/data/sw-library/.

## Compliance with industry standards

IBM Informix products are compliant with various standards.

IBM Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of IBM Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL Common Applications Environment (CAE) standards.

The IBM Informix Geodetic DataBlade® Module supports a subset of the data types from the *Spatial Data Transfer Standard (SDTS)—Federal Information Processing Standard 173*, as referenced by the document *Content Standard for Geospatial Metadata*, Federal Geographic Data Committee, June 8, 1994 (FGDC Metadata Standard).

## Syntax diagrams

Syntax diagrams use special components to describe the syntax for statements and commands.

*Table 6. Syntax Diagram Components*

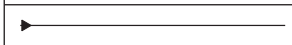
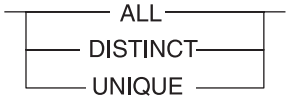
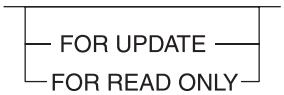
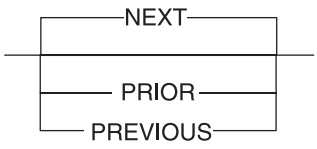
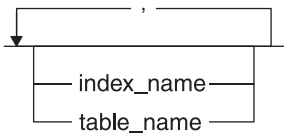
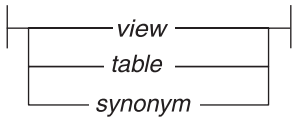
| Component represented in PDF | Component represented in HTML | Meaning |
|---|---|---|
|  | >>---------- | Statement begins. |
|  | ----------> | Statement continues on next line. |
|  | >---------- | Statement continues from previous line. |
|  | ---------->< | Statement ends. |
|  | --------SELECT---------- | Required item. |

*Table 6. Syntax Diagram Components (continued)*

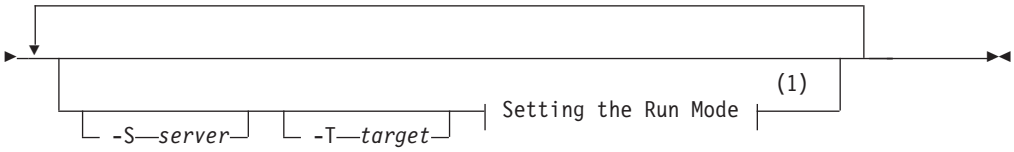
| Component represented in PDF | Component represented in HTML | Meaning |
|---|---|---|
|  | `--+-----------------+---`<br>`  '------LOCAL------'` | Optional item. |
|  | `---+-----ALL-------+---`<br>`   +--DISTINCT-----+`<br>`   '---UNIQUE------'` | Required item with choice. Only one item must be present. |
|  | `---+------------------+---`<br>`    +--FOR UPDATE-----+`<br>`    '--FOR READ ONLY--'` | Optional items with choice are shown below the main line, one of which you might specify. |
|  | `    .---NEXT---------.`<br>`----+----------------+---`<br>`    +---PRIOR--------+`<br>`    '---PREVIOUS-----'` | The values below the main line are optional, one of which you might specify. If you do not specify an item, the value above the line will be used as the default. |
|  | `  .-------,-----------.`<br>`  V                   |`<br>`---+-----------------+---`<br>`    +---index_name---+`<br>`    '---table_name---'` | Optional items. Several items are allowed; a comma must precede each repetition. |
|  | `>>-\| Table Reference \|-><` | Reference to a syntax segment. |
| Table Reference<br>  | `Table Reference`<br>`\|--+-----view--------+--\|`<br>`   +------table------+`<br>`   '----synonym------'` | Syntax segment. |

## How to read a command-line syntax diagram

Command-line syntax diagrams use similar elements to those of other syntax diagrams.

Some of the elements are listed in the table in Syntax Diagrams.

### Creating a no-conversion job

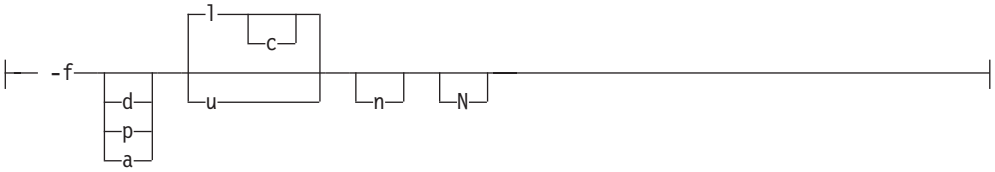




**Notes:**

1 See page Z-1

This diagram has a segment named “Setting the Run Mode,” which according to the diagram footnote is on page Z-1. If this was an actual cross-reference, you would find this segment on the first page of Appendix Z. Instead, this segment is shown in the following segment diagram. Notice that the diagram uses segment start and end components.

**Setting the run mode:**




To see how to construct a command correctly, start at the upper left of the main diagram. Follow the diagram to the right, including the elements that you want. The elements in this diagram are case-sensitive because they illustrate utility syntax. Other types of syntax, such as SQL, are not case-sensitive.

The Creating a No-Conversion Job diagram illustrates the following steps:

1. Type **onpladm create job** and then the name of the job.
2. Optionally, type **-p** and then the name of the project.
3. Type the following required elements:
   - **-n**
   - **-d** and the name of the device
   - **-D** and the name of the database
   - **-t** and the name of the table
4. Optionally, you can choose one or more of the following elements and repeat them an arbitrary number of times:
   - **-S** and the server name
   - **-T** and the target server name
   - The run mode. To set the run mode, follow the Setting the Run Mode segment diagram to type **-f**, optionally type **d**, **p**, or **a**, and then optionally type **l** or **u**.
5. Follow the diagram to the terminator.

## Keywords and punctuation

Keywords are words reserved for statements and all commands except system-level commands.

When a keyword appears in a syntax diagram, it is shown in uppercase letters. When you use a keyword in a command, you can write it in uppercase or lowercase letters, but you must spell the keyword exactly as it appears in the syntax diagram.

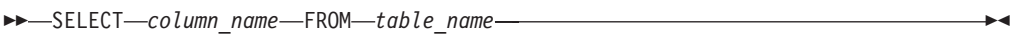
You must also use any punctuation in your statements and commands exactly as shown in the syntax diagrams.

### Identifiers and names

Variables serve as placeholders for identifiers and names in the syntax diagrams and examples.

You can replace a variable with an arbitrary name, identifier, or literal, depending on the context. Variables are also used to represent complex syntax elements that are expanded in additional syntax diagrams. When a variable appears in a syntax diagram, an example, or text, it is shown in *lowercase italic*.

The following syntax diagram uses variables to illustrate the general form of a simple SELECT statement.




When you write a SELECT statement of this form, you replace the variables *column_name* and *table_name* with the name of a specific column and table.

## How to provide documentation feedback

You are encouraged to send your comments about IBM Informix user documentation.

Use one of the following methods:

- Send e-mail to docinf@us.ibm.com.
- In the Informix information center, which is available online at http://www.ibm.com/software/data/sw-library/, open the topic that you want to comment on. Click the feedback link at the bottom of the page, fill out the form, and submit your feedback.
- Add comments to topics directly in the information center and read comments that were added by other users. Share information about the product documentation, participate in discussions with other users, rate topics, and more!

Feedback from all methods is monitored by the team that maintains the user documentation. The feedback methods are reserved for reporting errors and omissions in the documentation. For immediate help with a technical problem, contact IBM Technical Support. For instructions, see the IBM Informix Technical Support website at http://www.ibm.com/planetwide/.

We appreciate your suggestions.

# Chapter 1. Overview of IBM Data Server Provider for .NET

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

Two providers are included in the IBM Data Server Provider for .NET client package. These providers are sometimes referred to as the Common .NET Providers.

**The DB2 .NET Provider (IBM.Data.DB2.dll)**

With the DB2 .NET Provider your .NET applications can access the following database management systems:

- DB2® Database for Linux, UNIX, and Windows, Version 9.1, Version 9.5, Version 9.7, and Version 9.8
- DB2 Universal Database™ Version 8 for Windows, UNIX, and Linux
- DB2 Universal Database for z/OS® Version 8, Version 9, and Version 10 , through DB2 Connect™
- DB2 Universal Database Version 5 Release 4, Version 6 Release 1 and Version 7 Release 1 for iSeries®, through DB2 Connect (for IBM DB2 Version 9.7 Fix Pack 4 and higher versions)
- DB2 Universal Database Version 5 Release 4 and Version 6 Release 1 for iSeries, through DB2 Connect (for IBM DB2 Version 9.7 Fix Pack 3 and earlier versions)
- IBM Informix Version 11.10, Version 11.50, and Version 11.70

For more information about this provider, see the: IBM DB2 Database for Linux, UNIX, and Windows Information Center.

**The IDS .NET Provider (IBM.Data.Informix.dll)**

With the IDS .NET Provider your .NET applications can access the following database management systems:

- IBM Informix, Version 11.10 and Version 11.50

The remainder of these topics discuss the Common IDS .NET Provider.

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

In addition to the IBM Data Server Provider for .NET, the IBM Database Add-Ins for Visual Studio enable you to quickly and easily develop .NET applications for IBM data servers using Microsoft Visual Studio. You can also use the Add-Ins to create database objects such as indexes and tables, and develop server-side objects, such as stored procedures and user-defined functions.

## IBM Data Server Provider for .NET database system requirements for Informix

Before you install the IBM Data Provider for .NET, you must already have one of the following .NET Framework versions:

- .NET Framework Version 2.0
- .NET Framework Version 3.0

- .NET Framework Version 3.5
- .NET Framework Version 4.0

You can install the IBM Data Server Provider for .NET when you install IDS Client SDK (CSDK) or IBM Informix Connect. If you already have the IBM Informix .NET Provider installed on your computer, installing the IBM Data Server Provider for .NET will not effect the installation of the IBM Informix .NET Provider, which is also included in the CSDK and Informix Connect installation. You may use either or both of the .NET Providers. There are several Differences between .NET Providers.

## 32-bit and 64-bit support for ADO.NET applications

Versions of IBM Data Server Provider for .NET support 32-bit and 64-bit .NET applications. IBM Data Server Data Provider for .NET is shipped with Informix Dynamic Server Version 9.5 or later clients and servers.

The following table specifies the 32-bit and 64-bit support provided by combinations of IBM Data Server Provider for .NET and Microsoft .NET Framework.

*Table 1-1. 32-bit and 64-bit support provided by IBM Data Server Provider for .NET and Microsoft .NET Framework*

| IBM Data Server Provider for .NET and Microsoft .NET Framework versions | 32-bit support | 64-bit support |
|---|---|---|
| IBM Data Server Provider for .NET and Microsoft .NET Framework Version 2.0, Version 3.0, and Version 3.5 | Yes | Yes |

### 32-bit support in 64-bit for ADO.NET applications

Versions of IBM Data Server Provider for .NET support 32-bit .NET provider in the 64-bit client packages. When you install the 64-bit .NET provider package, both 32-bit and 64-bit providers are installed and configured.

## Differences between .NET Providers

There are runtime differences between the IBM Data Server .NET Provider and the IBM Informix .NET Provider. Knowing these differences will help you understand how each provider might affect existing applications and to select the provider that is right for your applications.

Both providers are available as part of the Client SDK (CSDK) for Informix.

For more information about these .NET Providers, see the *IBM Data Server Provider for .NET Programmer's Guide, Informix Edition* or the *IBM Informix .NET Provider Reference Guide*.

The following sections describe specific differences between the .NET providers.

### IBM Informix .NET Provider

The IBM Informix .NET Provider, sometimes referred to as the CSDK.NET provider, works with only the Informix database server and internally uses the SQLI protocol to communicate with Informix. The Informix .NET Provider is no

longer being enhanced for new .NET APIs.

## IBM Data Server .NET Provider

The IBM Data Server .NET Provider, sometimes referred to as the Common.NET provider, is the next generation of the .NET provider for application development. This .NET provider includes a number of capabilities, especially in the area of web application development, making it the preferred .NET provider for new client development.

The Data Server .NET Provider works with several different IBM data servers, including Informix, DB2 for Linux, UNIX, and Windows, and U2, and uses the DRDA® protocol. This provider is comprised of two different .NET provider assemblies that are designed for very specific application developer requirements:

**IBM.Data.DB2.dll**
Although the name of the provider indicates DB2, this is in fact the single .NET provider for IBM data servers including Informix and DB2. This is the recommended assembly for new application development for Informix Version 11.10 or later, and this is the preferred .NET provider.

**IBM.Data.Informix.dll**
This .NET provider assembly was created to help migrate existing .NET applications that were developed by using the Informix .NET Provider to use the latest DRDA protocol. This assembly works with Informix Version 11.10 or later. This assembly includes additional support for some of the legacy Informix client features and is targeted only for .NET application development for Informix.

## Comparison of supported features

There are several major differences between the features that are supported by the IBM Informix .NET provider and the IBM Data Server .NET Provider. Many of the features supported by the IBM Data Server .NET Provider are not supported by the IBM Informix .NET Provider.

The following table shows these differences:

*Table 1-2. Compare features supported by Informix .NET Provider and IBM Data Server .NET Provider*

| Feature | IBM Informix .NET Provider | IBM Data Server .NET Provider |
|---|---|---|
| Protocol support | SQLI | DRDA |
| Informix server support | Any supported Informix version | Informix v11.10 or later ONLY |
| LOB (BLOB & CLOB) column size limit | 4 TB | 2 GB |
| Support for .NET framework 3.0, 3.5 | No<br><br>Supports 1.1 and 2.0 frameworks. Limited VSAI windows application support. | Yes<br><br>Supports 2.0, 3.0, and 3.5 frameworks. |
| Support for LINQ (Entity Framework) | No | Yes |
| Silverlight and AJAX development support | No | Yes |

*Table 1-2. Compare features supported by Informix .NET Provider and IBM Data Server .NET Provider (continued)*

| Feature | IBM Informix .NET Provider | IBM Data Server .NET Provider |
|---|---|---|
| ASP.NET dynamic data support | No | Yes |
| ADO.NET Entity Data Modeling (EDM) support | No | Yes |
| Visual Studio Tools for Office (VSTO) development using EDM | No | Yes |
| VSAI support for Visual Studio 2008, Web application development support, WPF and WWF enhancements | No | Yes |
| VSAI Designers to create tables, procedures, functions and triggers, run procedures and functions | No | Yes |

## Unsupported keywords in the ConnectionString property

There are several connection string keywords that are not supported by the IBM Data Server .NET Provider. For example:

- DB_LOCALE
- CLIENT_LOCALE
- EXCLUSIVE

For a comprehensive list of the keywords that are supported by the IBM Data Server .NET Provider see: IfxConnection.ConnectionString Property.

## Connection pooling settings

In the IBM Informix .NET Provider, the Connection Lifetime attribute specifies how long a connection can remain open.

In the IBM Data Server .NET Provider, the Connection Lifetime attribute specifies the number of seconds that the connection can remain open and idle in the pool.

## Server keyword in the ConnectionString property

In the IBM Informix .NET Provider, the SERVER keyword should be used to specify the INFORMIXSERVER name as setup in the SetNet utility.

In the IBM Data Server .NET Provider, the SERVER keyword should be specified in the format `<hostname>:<port_number>`, where `<hostname>` is the machine name of the instance and `<port_number>` is the port on which the instance is listening.

## Database keyword in the ConnectionString property

With the IBM Informix .NET Provider, you can connect to a server without specifying a database. With the IBM Data Server .NET Provider, `database` is a mandatory attribute.

The IBM Informix .NET Provider allows specifying the value for the Database attribute in the format `dbname@server`. The IBM Data Server .NET Provider does not support this format.

## String values in quotation marks in the ConnectionString property

The IBM Informix .NET Provider supports database, server, and password keyword values in quotation marks.

Applications that use the IBM Data Server .NET Provider can specify values in quotation marks for the database and password keywords only when specifying an alias for a catalog connection.

## IfxConnection.ServerType property

For this property, the IBM Informix .NET Provider returns the value *Informix*. The IBM Data Server .NET Provider returns the value that is received from the database server to which it is connected. For example, the provider connected to an Informix database server installed on a UNIX 64-bit system might return *IDS/UNIX64*.

## Unsupported data types in SQL statements

The IBM Data Server .NET Provider does not allow access to the LIST, MULTISET, SET,or ROW data types because the Informix DRDA server does not support these types.

With the IBM Informix .NET Provider, you can select and return the values from the LIST, MULTISET, and ROW data types.

## BYTE and TEXT data types

The IBM Data Server .NET Provider maps both Byte and BLOB data types to the IfxBlob data type. When binding an IfxBlob object as a parameter, applications must use the `::byte` clause after the parameter marker. This clause indicates that the IfxBlob value corresponds to a byte column. Without the `::byte` clause, a conversion error is returned.

The IBM Data Server .NET Provider maps both Text and CLOB data types to the IfxClob data type. When binding an IfxClob object as a parameter, applications must use the `::text` clause after the parameter marker. This clause indicates that the IfxClob value corresponds to a text column. Without the `::text` clause, a conversion error is returned.

```
DB2Parameter clobParam;

clobParam.IfxType = IfxType.Clob;
...
cmd.CommandText = "CALL textSP(?::text)";
---
cmd.ExecuteNonQuery();
```

## Supported data types

The following table compares the more unusual Informix data types and how each .NET provider supports those data types.

*Table 1-3. Differences between .NET provider support for unusual Informix data types*

| Informix Data Type | IBM Informix .NET Provider | IBM Data Server .NET Provider |
|---|---|---|
| CLOB, BLOB | Supported | Limited Support. The size limitation is 2 GB. |
| Collection | Supported | Not supported |
| Interval DayToFraction | Supported. This type must be read as a string. | Not supported |
| IntervalMonth | Supported | Not supported |
| LIST | Supported | Not supported |
| MONEY | Supported | Money will be treated as decimal data type. |
| MULTISET | Supported | Not Supported |
| ROW | Supported | Not Supported |
| SET | Supported | Not Supported |

## Comparing classes and structures

The following table compares the differences between the .NET provider support for the data type classes and structures support.

*Table 1-4. Differences between .NET provider support for data type classes and structures*

| Class / Structure | IBM Informix .NET Provider | IBM Data Server .NET Provider |
|---|---|---|
| IfxBlob | Supported | Limited Support. Some methods return a NotImplemented exception.<br><br>For more information about the supported methods see: IfxBlobClass. |
| IfxClob | Supported | Limited Support. Some methods return a NotImplemented exception.<br><br>For more information about the supported methods see: IfxClobClass. |
| IfxTimeSpan | Supported | Not Supported |
| IfxMonthSpan | Supported | Not Supported |
| IfxSmartLOBLocator | Supported | Not Supported |
| IfxSmartLOB | Supported | Not Supported |

*Table 1-4. Differences between .NET provider support for data type classes and structures (continued)*

| Class / Structure | IBM Informix .NET Provider | IBM Data Server .NET Provider |
|---|---|---|
| IfxDecimal | Supported | Limited Support. Some methods return a NotImplemented exception.<br><br>For more information about the supported methods see: IfxDecimalClass. |
| IfxDateTime | Supported | Limited Support.<br><br>DRDA client supports 6 digits in the fractional part of DateTime. The Informix server supports 5 digits of the fractional value. Because of this difference, the least significant digit of the fraction will be truncated. A DateTime value read from the Informix server has one zero added to the least significant digit to match the DRDA format. This change affects applications that access the IfxDateTime type from the IBM Informix .NET Provider. |

## Stored procedure differences

With the IBM Informix .NET Provider applications can read the return value of a stored procedure or function as a ReturnValue parameter.

With the IBM Data Server .NET Provider applications by default can read the return value of a stored procedure or function as a data reader. To read it is a ReturnValue parameter, applications must set the ResultSetAsReturnValue parameter to true in the IfxCommand or IfxConnection object. To read it is a ReturnValue parameter, applications must set the ResultSetAsReturnValue property to true in the IfxCommand or IfxConnection object.

## IfxCommand.ExecuteScalar() method

The following query is a sample SELECT query.

```
Cmd.CommandText = "SELECT COUNT(*) FROM Tab";
```

The data type that is returned depends on the .NET provider that you use:

**IBM Informix .NET Provider**

When you use the IBM Informix .NET Provider, this query returns the count as decimal type.

```
Decimal count = (Decimal) Cmd.ExecuteScalar();
```

**IBM Data Server .NET Provider**

When you use the IBM Data Server .NET Provider, this query returns the count as Int32.

```
Int32 count = (Int32) Cmd.ExecuteScalar();
```

## IfxDataReader.GetString() method

The IBM Data Server .NET Provider does not allow the IfxDataReader.GetString() method if the underlying value is null. A null value results in an InvalidCast exception. This behavior is in contrast to the IBM Informix .NET Provider, which allows a null value and returns an empty string.

## Error messages

Some error messages received by the IBM Data Server .NET Provider might differ from those received by the IBM Informix .NET Provider. For example, the IBM Informix .NET Provider has tags such as `[Informix .NET provider]` in the error messages. The IBM Data Server .NET Provider has just `[IBM]` in the error messages. Additionally, the error codes might be different for the same type of error.

# Chapter 2. IBM data server clients and drivers overview

This topic outlines information available about clients and drivers, and provides links to further details. This topic will help you to perform the following tasks:

1. Choose the appropriate IBM data server client or driver to enable connections between your system and remote databases.
2. Choose the most suitable method for installing your client or driver.
3. Complete the steps and address the considerations needed to set up a client or driver.

## Connection options

Options for connecting a system to a remote database include various IBM data server clients and drivers. The options available depend on whether the system connecting to the remote database is:

- An application located on a business user's machine or an application server
- An application development workstation
- A database administrator workstation

There are additional options to consider if you need to also connect to midrange or mainframe databases.

## IBM data server client and driver types

The following lists the IBM data server clients and drivers:

- IBM Data Server Driver Package
- IBM Data Server Client
- IBM Data Server Runtime Client
- IBM Data Server Driver for JDBC and SQLJ.
- IBM Data Server Driver for ODBC and CLI

The IBM Data Server Driver is a lightweight solution and the recommended best practice package for end user code deployment. It provides robust runtime support for applications using ODBC, CLI, .NET, OLE DB, PHP, Ruby, JDBC, or SQLJ without the need of installing Data Server Runtime Client or Data Server Client. The IBM Data Server Driver for ODBC and CLI solution is designed mainly for independent software vendor (ISV) deployments.

It is also recommended that the IBM Data Server Driver Package be installed first. It can then be configured to work in conjunction with DB2 Connect.

In addition, a separate product, DB2 Connect Personal Edition, includes all the functionality of IBM Data Server Client plus the capability to connect to midrange and mainframe databases.

DB2 Connect capability can be added to any client or driver.

### Installation methods

The common method for installing a client or driver is to run the installation program provided on a product DVD.

The IBM Data Server Driver Package on Windows can be installed by following the prompts. There is no installation program for IBM Data Server Driver for ODBC and CLI or for IBM Data Server Driver Package on Linux and UNIX. You must install the driver manually.

Other installation methods are also available. Some methods are designed to automate the deployment of large numbers of clients. Other methods use various Windows operating system capabilities. For example, on Windows operating systems, you can use merge modules to embed the functionality of Data Server Runtime Client or IBM Data Server Driver Package in your application.

### Setting up a client or driver

After you decide which client to use, set up the client by performing the following steps:

1. Ensure that system prerequisites are satisfied.
2. Perform the installation.

For systems where a Version 8 client or a DB2 Version 9 client already exists, consider whether to upgrade the existing client to a Version 9.7 Data Server Client, or, keep the pre-Version 9.7 client and install the Version 9.7 Data Server Client as an additional client. Upgrading to a Version 9.7 client is strongly recommended. Running multiple copies of the client packages is for advanced users only.

**Note:** The option to upgrade and replace the existing client applies to Data Server Client only.

## IBM data server client types

There are several types of IBM data server clients and drivers available. Each provides a particular type of support.

The following lists the IBM data server client and driver types available to you:

- IBM Data Server Driver for JDBC and SQLJ
- IBM Data Server Driver for ODBC and CLI
- IBM Data Server Driver Package
- IBM Data Server Runtime Client
- IBM Data Server Client

The DB2 Connect Personal Edition product includes all the functionality of IBM Data Server Client and connects to midrange and mainframe databases. DB2 Connect capability can be added to any client or driver.

Each IBM data server client and driver provides a particular type of support:

- For Java applications only, use IBM Data Server Driver for JDBC and SQLJ.
- For applications using ODBC or CLI only, use IBM Data Server Driver for ODBC and CLI. (Also referred to as cli driver.)

- For applications using ODBC, CLI, .NET, OLE DB, PHP, Ruby, JDBC, or SQLJ, use IBM Data Server Driver Package.
- For applications using DB2CI, use IBM Data Server Client.
- If you need DB2 Command Line Processor Plus (CLPPlus) support, use IBM Data Server Driver Package.
- To have command line processor (CLP) support and basic client support for running and deploying applications, use IBM Data Server Runtime Client.
- To have support for database administration, and application development using an application programming interface (API), such as ODBC, CLI, .NET, or JDBC, use IBM Data Server Client.

## IBM Data Server Driver for JDBC and SQLJ

IBM Data Server Driver for JDBC and SQLJ is the default driver for Java stored procedures and user-defined functions. This driver provides support for client applications and applets that are written in Java using JDBC to access local or remote servers, and SQLJ for embedded static SQL in Java applications. This driver is a prerequisite for Optim™ pureQuery Runtime, which provides static support for Java, enables optimized data access using the pureQuery API, and is supported by a full integrated development environment (IDE) for Java database application development using Optim Development Studio. (Both Optim products are available separately.)

## IBM Data Server Driver for ODBC and CLI

Data Server Driver for ODBC and CLI is a lightweight deployment solution designed for independent software vendors (ISV) deployments. This driver, also referred to as cli driver, provides runtime support for applications using ODBC API, or CLI API without need of installing the Data Server Client or the Data Server Runtime Client. This driver is available only as a tar file, not as an installable image. Messages are reported only in English.

The IBM Data Server Driver for ODBC and CLI provides:

- runtime support for the CLI API;
- runtime support for the ODBC API;
- runtime support for the XA API;
- database connectivity;
- support for DB2 Interactive Call Level Interface (**db2cli**);
- LDAP Database Directory support; and
- tracing, logging, and diagnostic support.

Register Data Server Driver for ODBC and CLI with the Microsoft ODBC driver manager using the `db2oreg1.exe` utility.

## IBM Data Server Driver Package

IBM Data Server Driver Package is a lightweight deployment solution that provides runtime support for applications using ODBC, CLI, .NET, OLE DB, PHP, Ruby, JDBC, or SQLJ without the need of installing Data Server Runtime Client or Data Server Client. This driver has a small footprint and is designed to be redistributed by independent software vendors (ISVs), and to be used for application distribution in mass deployment scenarios typical of large enterprises.

The IBM Data Server Driver Package capabilities include:

- The DB2 Command Line Processor Plus (CLPPlus) for dynamically creating, editing, and running SQL statements and scripts.
- Support for applications that use ODBC, CLI, PHP, or Ruby to access databases.
- Support for client applications and applets that are written in Java using JDBC, and for embedded SQL for Java (SQLJ).
- IBM Informix support for .NET, PHP, and Ruby.
- Support for running embedded SQL applications. No precompiler or bind capabilities are provided.
- Application header files to rebuild the PHP, Ruby, Python, and Perl drivers. The Python and Perl drivers are not available in IBM Data Server Driver Package; however, you can download and build these drivers using the header files.
- Support for DB2 Interactive Call Level Interface (**db2cli**).
- Support for the for DRDA traces (**db2drdat**).
- On Windows operating systems, IBM Data Server Driver Package also provides support for applications that use .NET or OLE DB to access databases. In addition, this is available as an installable image, and merge modules are available to allow you to easily embed the driver in a Windows Installer-based installation.

## IBM Data Server Runtime Client

The IBM Data Server Runtime Client provides a way to run applications on remote databases. GUI tools are not shipped with the IBM Data Server Runtime Client.

Capabilities include:

- The DB2 command line processor (CLP) for issuing commands. The CLP also provides a basic way to perform remote administration of servers.
- Base client support to handle database connections, SQL statements, XQuery statements, and commands.
- Support for common database access interfaces: JDBC, ADO.NET, OLE DB, ODBC, Command Line Interface (CLI), PHP, and Ruby. This support includes drivers and capabilities to define data sources. For example, for ODBC, installing an IBM data server client installs the ODBC driver and registers the driver. Application developers and other users can use the Windows ODBC Data Source Administrator tool to define data sources.
- Lightweight Directory Access Protocol (LDAP) exploitation.
- Support for common network communication protocols: TCP/IP, and Named Pipe.
- Support for installing multiple copies of a client on the same computer. These copies can be the same or different versions.
- License terms that allow free redistribution of IBM Data Server Runtime Client with your application.
- smaller deployment footprint compared to that of the full IBM Data Server Client in terms of installation image size and disk space required.
- A catalog that stores information for connecting to databases and servers.
- Packing advantages on Windows operating systems: You can package the client with your application to provide connectivity for that application. Also, the client is available as Windows Installer merge modules that enable you to include the RTCL DLL files in your application installation package. This approach also enables you to include only the parts of the client that you need with your application.

- IBM Informix support for PHP, Ruby, .NET, and JDBC

### IBM Data Server Client

IBM Data Server Client includes all the functionality of IBM Data Server Runtime Client, plus functionality for database administration, application development, and client/server configuration.

Capabilities include the following:

- The ability to prune the IBM Data Server Client image to reduce the installation image size on the Windows operating system.
- The configuration Assistant to assist with cataloging databases and configuring the database server.
- Control Center and other graphical tools for database implementation and for database administration. These tools are available for versions of Windows on x86 (32-bit only), Windows on x64 (AMD64/EM64T), Linux on x86, and Linux on AMD64/EM64T (x64).
- First Steps documentation for new users.
- Visual Studio tools
- Application header files
- Precompilers for various programming languages
- Bind support
- Samples and tutorials
- IBM Informix support for PHP, Ruby, .NET, JCC, and JDBC

## IBM Data Server Driver restrictions

The IBM Data Server Driver Package simplifies application deployment. However, certain restrictions apply.

The following restrictions apply to IBM Data Server Driver Package:

- No other database product can be installed in the same path if IBM Data Server Driver Package is already installed.
- On Windows operating systems, you can install a maximum of 16 copies of IBM Data Server Driver Package.
- To connect to a z/OS server or a System i® server, you must register a DB2 Connect license key. (Retrieve the license file from your Passport Advantage® distribution, for example `db2conpe.lic`, then copy the license file to the license directory under the directory where the driver was installed.)
- XA connections against a z/OS server are supported. However, XA connections against a System i server are not supported.
- If you use the configuration file `db2dsdriver.cfg` to specify aliases, the following entries must contain a value:
  - <dsncollection> entries (alias, name, host, and port)
  - <database> entries (name, host, port).

  These entries must be specified and cannot be empty.
- The CLI/ODBC configuration keyword **DBNAME** is not supported.
- The CLI LOAD utility statement attribute, sql_attr_use_load_api, is not supported.

### Functionality supported with restrictions

- There is no local database directory. Lightweight Directory Access Protocol (LDAP) is supported, but the LDAP cache is not saved to disk.
- Runtime support for embedded SQL is available with the following restrictions:
  - Support is runtime only; there is no **PREP** or **BIND** capability.
  - Support is available for DDL and DML SQL statements that are called from EXEC SQL or cursors.
  - When reading the configuration file `db2dsdriver.cfg`, embedded SQL applications can only access connection information (for example, database name, host name, and port number).
  - Sysplex capabilities are not supported.
  - The following APIs are not supported:
    - APIs for loading (db2Load and db2LoadQuery), exporting (db2Export), and importing (db2Import) data.
    - API for getting the current user authorities (sqluadau).
    - API for forcing users and applications off the system (sqlefrce).
- Executing an embedded compound statement with sub-statements is not supported.

### Functionality not supported

- DB2 Command Line Processor (CLP)
- administrative APIs
- installation program
- The CLIENT type authentication is not supported by the IBM Data Server Driver for ODBC and CLI and by the IBM Data Server Driver Package.

### Known limitations

For information on short term limitations, go to http://www.ibm.com/developerworks/wikis/display/DB2/IBM+Data+Server+Driver+Limitations

## db2dsdriver configuration file

The `db2dsdriver.cfg` configuration file contains database directory information and client configuration parameters in a human-readable format.

The `db2dsdriver.cfg` configuration file is an ASCII file that contains various keywords and values that can be used to make a connection to a supported database through ODBC, CLI, .NET, OLE DB, or open source (PHP or Ruby) applications by using keywords. The keywords are associated with the database *alias name*, and affect all the applications that access the database. You can also use this configuration file to specify Sysplex-related settings such as Sysplex workload balancing.

This configuration file can be used with the following data server drivers:

- cli driver:
  - IBM Data Server Driver for ODBC and CLI
- ds driver:
  - IBM Data Server Driver Package

    The `db2dsdriver.cfg` file can be used with all clients in the IBM Data Server Driver Package except the IBM Data Server Driver for JDBC and SQLJ.

- IBM data server clients:
  - IBM Data Server Client
  - IBM Data Server Runtime Client

  In Version 9.7 Fix Pack 3 and later, CLI and open source applications that use IBM Data Server Client or IBM Data Server Runtime Client can use the configuration file to retrieve all database connection information and properties.

  Other applications that use IBM Data Server Client or IBM Data Server Runtime Client, such as .NET applications or applications that use embedded SQL, can retrieve only Sysplex-related settings from the `db2dsdriver.cfg` configuration file. Database connection information and properties should come from a different source, such as the database catalog, a connection string, the `db2cli.ini` initialization file, or .NET object properties.

You do not need to create and populate the `db2dsdriver.cfg` configuration file for these drivers. They can function without this configuration file. However, instead of specifying information about the database name, host, port, and configuration parameters in your applications, you can use the configuration file to define aliases.

The various keywords, values, and specific database connection settings that are specified in the configuration file apply to all ODBC, CLI, .NET or open source application connections made to that database or alias.

In Version 9.7 Fix Pack 3 and later, the `db2dsdriver.cfg` configuration file is not shipped. Instead, the `db2dsdriver.cfg.sample` sample configuration file is shipped to help you get started. Use the contents of the `db2dsdriver.cfg.sample` file to create a `db2dsdriver.cfg` file in the same location as the sample configuration file. The location of the sample configuration file depends on your driver type and platform.

For IBM Data Server Client, IBM Data Server Runtime Client, or IBM Data Server Driver Package, the configuration file is created in one of the following paths:

- On AIX®, HP-UX, Linux, or Solaris operating systems: *instance_path*`/cfg`
- On Windows XP and Windows Server 2003: `C:\Documents and Settings\All Users\Application Data\IBM\DB2\`*driver_copy_name*`\cfg`
- On Windows Vista and Windows Server 2008: `C:\ProgramData\IBM\DB2\`*driver_copy_name*`\cfg`

For example, if you use IBM Data Server Driver Package for Windows XP, and the data server driver copy name is `DSD_COPY`, then the `db2dsdriver.cfg.sample` file is created in the `C:\Documents and Settings\All Users\Application Data\IBM\DB2\DSD_COPY\cfg` directory.

For IBM Data Server Driver for ODBC and CLI, the configuration file is created in one of the following paths:

- On AIX, HP-UX, Linux, or Solaris operating systems: *instance_path*`/cfg`
- On Windows XP and Windows Server 2003: `C:\Documents and Settings\All Users\Application Data\IBM\DB2\`*driver_installation_path*`\cfg`

  where *driver_installation_path* is the file path where the driver is installed, with each directory separated by an underscore (_) instead of a backslash (\).
- On Windows Vista and Windows Server 2008: `C:\ProgramData\IBM\DB2\`*driver_installation_path*`\cfg`

where *driver_installation_path* is the file path where the driver is installed, with each directory separated by an underscore (_) instead of a backslash (\).

For example, if you use IBM Data Server Driver for ODBC and CLI for Windows Vista, and the driver is installed in the `C:\IBMDB2\CLIDRIVER\V97FP3` directory, then the `db2dsdriver.cfg.sample` file is created in the `C:\ProgramData\IBM\DB2\C_IBMDB2_CLIDRIVER_V97FP3\cfg` directory.

If you use a copy of the `db2dsdriver.cfg` file from Version 9.7 Fix Pack 2 or earlier on Windows, the file is in a different location. You can keep the copy of the `db2dsdriver.cfg` file in that location, but the location might not be valid in future releases.

The `db2dsdriver.cfg` configuration file can be copied and edited manually. After editing the file, you must restart the application for the changes to take effect.

If you have an IBM Data Server Runtime Client or IBM Data Server Client, you can copy the existing database directory information into the `db2dsdriver.cfg` configuration file by using the **db2dsdcfgfill** command. When you run this command, the configuration file is populated based on the contents of the local database directory, node directory, and Database Connection Services (DCS) directory of a specific database manager instance.

If you have an IBM Data Server Driver for ODBC and CLI client, you can create the `db2dsdriver.cfg.sample` and `db2cli.ini.sample` sample configuration files by using the `db2oreg1.exe` utility.

IBM Data Server Client and IBM Data Server Runtime Client can catalog remote databases locally. In Version 9.7 Fix Pack 3 and later, you can define client parameters for the databases that are cataloged. IBM Data Server Client and IBM Data Server Runtime Client derive database, host, and port information from the catalog directory and use that information to locate the corresponding entry in the `db2dsdriver.cfg` configuration file.

When IBM Data Server Client or IBM Data Server Runtime Client connects to a database, the client identifies database details by looking at a series of sources in the following order of precedence:

1. The connection string.
2. The `db2cli.ini` file (not applicable to .NET providers).
3. The `db2dsdriver.cfg` file.

The new order of precedence for .NET is as follows:

1. The connection string.
2. Database and Node directories (for IBM Data Server Runtime Client)
3. The `db2dsdriver.cfg` file.

The following restrictions apply to the `db2dsdriver.cfg` configuration file:

- The configuration file cannot contain multiple identical entries for a database with the following properties: database name, server name and port number. In addition, the configuration file cannot contain multiple identical database alias entries.
- The <dsncollection> entries (alias, name, host, and port) and the <database> entries (name, host, port) must contain a value.
- If multiple parameters are defined on a single line, they are ignored.

This configuration file is an XML file that is based on the db2dsdriver.xsd schema definition file. In Version 9.7 Fix Pack 3, the db2dsdriver.cfg configuration file supports a consistent set of XML tags that are in lower case and do not include underscores (_). You can use tags that are in mixed case or include underscores, but these tags might not be valid in future releases.

The sample configuration file contains the following sections:

- the data source name section contained within the <dsncollection> tags,
- the database information section contained within the <databases> tags,
- the global attributes section contained within the <parameters> tags.

```
<configuration>
 <dsncollection>
  <dsn alias="alias1" name="name1" host="server1.net1.com" port="50001"/>
  <!-- Long aliases are supported -->
  <dsn alias="longaliasname2" name="name2" host="server2.net1.com" port="55551">
   <parameter name="Authentication" value="Client"/>
  </dsn>
 </dsncollection>
 <databases>
  <database name="name1" host="server1.net1.com" port="50001">
   <parameter name="CurrentSchema" value="OWNER1"/>
   <wlb>
    <parameter name="enableWLB" value="true"/>
    <parameter name="maxTransports" value="50"/>
   </wlb>
   <acr>
    <parameter name="enableACR" value="true"/>
   </acr>
  </database>
  <!-- Local IPC connection -->
  <database name="name3" host="localhost" port="0">
   <parameter name="IPCInstance" value="DB2"/>
   <parameter name="CommProtocol" value="IPC"/>
  </database>
 </databases>
 <parameters>
  <parameter name="GlobalParam" value="Value"/>
 </parameters>
</configuration>
```

# db2dsdcfgfill - Create configuration file db2dsdriver.cfg

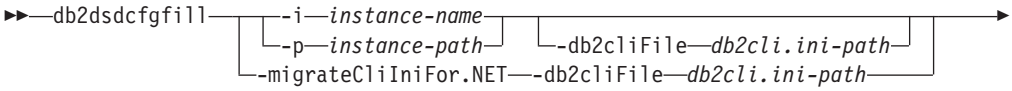
Creates and populates the configuration file db2dsdriver.cfg automatically.

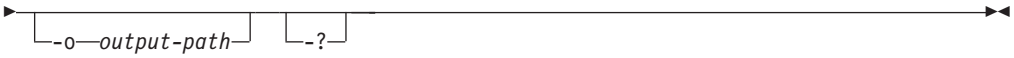
## Description

1 After installing IBM Data Server Driver Package, you can run this command to automatically create and populate the db2dsdriver.cfg configuration file in a human-readable format.

The **db2dsdcfgfill** command copies the existing database directory information from either the existing IBM Data Server Client or IBM Data Server Runtime Client, and copies the information to the db2dsdriver.cfg configuration file.

## Command syntax






## Command parameters

**-i** *instance-name*
The name of the database manager instance whose database directory, node directory, and Database Connection Services (DCS) directory will be used as input by the **db2dsdcfgfill** command.

Cannot be used in combination with **-p** or **-migrateCliIniFor.NET**.

**-p** *instance-path*
The full path of the database manager instance directory under which the system database directory, node directory, and DCS directory are located.

Cannot be used in combination with **-i** or **-migrateCliIniFor.NET**.

**-migrateCliIniFor.NET**

Copies certain entries from `db2cli.ini` into the `db2dsdriver.cfg` file. This option is for Microsoft Windows systems only. Only the following keywords will be migrated:

- txnisolation
- connecttimeout
- currentschema

The keywords will be migrated in the following manner:

- Entries in the common section of the `db2cli.ini` file will be copied to the global section of the `db2dsdriver.cfg` file.
- Entries that have a database name, host name, and port information will be copied to the database section.
- Entries for cataloged databases will be copied to the DSN section.

Cannot be used in combination with **-i** or **-p**.

**Note:** For .NET applications and applications that use embedded SQL, IBM Data Server Client and IBM Data Server Runtime Client can use the `db2dsdriver.cfg` file to retrieve only Sysplex-related settings.

**-db2cliFile** *db2cli.ini-path*
The full path of the `db2cli.ini` file. This option is for Microsoft Windows systems only.

**-o** *output-path*
The path where the **db2dsdcfgfill** command creates the `db2dsdriver.cfg` configuration file.

If you do not specify a value for this parameter, and you have a copy of the `db2dsdriver.cfg` file from Version 9.7 Fix Pack 2 or earlier, that copy is replaced. Otherwise, if you do not specify a value for this parameter, the `db2dsdriver.cfg` configuration file is created in a directory that depends on your driver type and platform. For information about the location of the `db2dsdriver.cfg` file, see the topic about that file.

**-?** Displays usage information.

### Usage notes

When you run the **db2dsdcfgfill** command, if a db2dsdriver.cfg configuration file already exists in the output directory, the existing db2dsdriver.cfg configuration file is overwritten. However, when the **-migrateCliIniFor.NET** option is used with an existing db2dsdriver.cfg file, it will merge the information into the existing file instead.

## Copying existing database directory information into the db2dsdriver configuration file

You can populate the db2dsdriver.cfg configuration file with existing database directory information.

You must have an existing Version 9.7 IBM Data Server Client or IBM Data Server Runtime Client installed.

The db2dsdriver.cfg configuration file configures the behavior of DB2 CLI, ODBC, open source, or .NET applications by using keywords. The keywords are associated with the database alias name, and affect all the applications that access the database.

If you have an IBM Data Server Client or IBM Data Server Runtime Client, you can copy the existing database directory information into the db2dsdriver.cfg configuration file by using the **db2dsdcfgfill** command. Using this command, the configuration file is populated based on the contents of the local database directory, node directory, and Database Connection Services (DCS) directory of a specific database manager instance.

**Restrictions**

None.

To copy existing database directory information from an IBM Data Server Client or IBM Data Server Runtime Client into the db2dsdriver.cfg configuration file:

Enter the **db2dsdcfgfill** command. For example, db2dsdcfgfill -i *instance_name* -p *instance_path* -o *output_path*. The parameter **-o** *output-path* indicates the path where the db2dsdriver.cfg configuration file is created. For information about the location of the db2dsdriver.cfg file, see the topic about that file.

# IBM Data Server Driver configuration keywords

The IBM Data Server Driver configuration keywords can be used to configure connections to supported databases through ODBC, CLI, .NET, OLE DB, or open source (PHP or Ruby) applications. The keywords are associated with the database alias name, and affect all the applications that access that database. These keywords and their corresponding values are stored in the db2dsdriver.cfg file.

## AllowDeferredPrepare IBM Data Server Driver configuration keyword

Minimizes network flow by combining the PREPARE request with the corresponding execute request.

**Equivalent CLI keyword**
DeferredPrepare

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="AllowDeferredPrepare" value="0 | 1"/>
```

**Default setting:**
The prepare request will be delayed until the execute request is sent.

**Equivalent statement attribute:**
SQL_ATTR_DEFERRED_PREPARE

**Usage notes:**

Defers sending the PREPARE request until the corresponding execute request is issued. The two requests are then combined into one command/reply flow (instead of two) to minimize network flow and to improve performance.

- 0 = SQL_DEFERRED_PREPARE_OFF. The PREPARE request will be executed the moment it is issued.
- 1 = SQL_DEFERRED_PREPARE_ON (default). Defer the execution of the PREPARE request until the corresponding execute request is issued.

  If the target DBMS does not support deferred prepare, the client disables deferred prepare for that connection.

**Note:** When deferred prepare is enabled, the row and cost estimates normally returned in the SQLERRD(3) and SQLERRD(4) of the SQLCA of a PREPARE statement may become zeros. This may be of concern to users who want to use these values to decide whether or not to continue the SQL statement.

## AllowedCursorTypes IBM Data Server Driver configuration keyword

Specifies which cursor types are permitted.

**Equivalent CLI keyword**
CursorTypes

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="AllowedCursorTypes" value="0 | 1 | 2 | 3 | 4 | 5 |
6 | 7"/>
```

**Default setting:**
Forward-only, static, keyset-driven, and dynamic cursors are supported if the server supports them.

**Usage notes:**

The AllowedCursorTypes keyword is a bitmask that indicates what types of cursors an application can open:

- 0x0 - forward-only (can always be opened)
- 0x1 - static
- 0x2 - keyset-driven
- 0x4 - dynamic

For example,

- to prevent applications from opening dynamic scrollable cursors, set AllowedCursorTypes to 3.
- to allow applications to open only nonscrollable cursors, set AllowedCursorTypes to 0.

This keyword only affects calls made to the following functions:

- `SQLBulkOperations()`
- `SQLExecDirect()`
- `SQLExecute()`
- `SQLFetchScroll()`
- `SQLPrepare()`
- `SQLSetPos()`

## AllowGetDataLobReaccess IBM Data Server Driver Configuration Keyword

Specifies whether the application can call `SQLGetData()` for previously accessed LOB columns when querying database servers that support Dynamic Data Format.

**Equivalent CLI keyword**
AllowGetDataLOBReaccess

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="AllowGetDataLobReaccess" value="0 | 1"/>
```

**Default setting:**
Do not allow calls to `SQLGetData()` for previously accessed LOB columns when querying database servers that support Dynamic Data Format.

**Usage notes:**
This keyword only affects connections to database servers that support Dynamic Data Format, also known as progressive streaming. The default setting of 0 does not allow applications to call `SQLGetData()` for previously accessed LOB columns. Specify 1 to allow applications to call `SQLGetData()` for previously accessed LOB columns.

Note that when the keyword is set to 1 to allow re-access to LOB columns, some resources on the server might not be freed upon completion of `SQLGetData()`.

If the server does not support Dynamic Data Format, this keyword has no effect and calls to `SQLGetData()` for previously accessed LOB columns are allowed.

## AllowInterleavedGetData IBM Data Server Driver configuration keyword

Specifies whether the application can call `SQLGetData()` for previously accessed LOB columns and maintain the data offset position from the previous call to `SQLGetData()` when querying data servers that support Dynamic Data Format.

**Equivalent CLI keyword**
AllowInterleavedGetData

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="AllowInterleavedGetData" value="FALSE  | TRUE"/>
```

**Default setting:**

Do not allow calls to SQLGetData() for previously accessed LOB columns when querying database servers that support Dynamic Data Format.

**Usage notes:**

This keyword affects only connections to database servers that support Dynamic Data Format, also known as progressive streaming. The default setting of FALSE does not allow applications to call SQLGetData() for previously accessed LOB columns. Specify TRUE to allow applications to call SQLGetData() for previously accessed LOB columns and start reading LOB data from where the application stopped reading during the previous read.

Note that when the keyword is set to TRUE to allow re-access to LOB columns, some resources on the server might not be freed upon completion of SQLGetData().

If the server does not support Dynamic Data Format, this keyword has no effect, and calls to SQLGetData() for previously accessed LOB columns are allowed.

A similar keyword exists called AllowGetDataLOBReaccess that allows applications to call SQLGetData() for previously accessed LOB columns. However, if the AllowGetDataLOBReaccess keyword is used, data position and offset information is not maintained. When the LOB column is re-accessed after interleaving, SQLGetData() starts reading data from the beginning for that LOB data column. If both AllowGetDataLOBReaccess and AllowInterleavedGetData are set, the AllowInterleavedGetData setting takes precedence over AllowGetDataLOBReaccess.

## AlternateZOSSysSchema IBM Data Server Driver configuration keyword

Sets an alternative schema to be searched in place of the SYSIBM schema.

**Equivalent CLI keyword**

SysSchema

**Equivalent IBM Data Server Provider for .NET connection string keyword**

N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="AlternateZOSSysSchema" value="alternative schema"/>
```

**Default setting:**

The default table qualifier name used when querying DB2 for z/OS is SYSIBM.

**Usage notes:**

This option indicates an alternative schema, or table qualifier, to be searched in place of the SYSIBM schema when the DB2 CLI and ODBC Catalog Function calls are issued to obtain system catalog information from DB2 for z/OS.

Using this new schema name, the system administrator can define a set of views, or a copies of the tables, consisting of a subset of the rows for system catalog tables such as:

- SYSIBM.SYSCOLAUTH

- SYSIBM.SYSCOLUMNS
- SYSIBM.SYSDATATYPES
- SYSIBM.SYSFOREIGNKEYS
- SYSIBM.SYSINDEXES
- SYSIBM.SYSKEYS
- SYSIBM.SYSKEYCOLUSES
- SYSIBM.SYSPARMS
- SYSIBM.SYSRELS
- SYSIBM.SYSROUTINES
- SYSIBM.SYSTABAUTH
- SYSIBM.SYSTABCONST
- SYSIBM.SYSTABLES
- SYSIBM.SYSSYNONYMS

For example, if the set of views, or a copies tables, for the system catalog tables is in the ACME schema, then the view (or copy of the table) for SYSIBM.SYSTABLES is ACME.SYSTABLES; and AlternateZOSSysSchema should be set to ACME.

For applications that automatically query the system catalogs for all table names, defining and using limited views of the system catalog tables reduces the number of tables listed by the application. This can reduce the time it takes for the application to query table information since a subset of table names is returned.

Defining and using copies of the system catalog tables, with the same indexes defined on the copy as those defined on the system table, can reduce the time it takes for applications to query the database.

The SchemaList, TableType and DBName keywords can be used in conjunction with the AlternateZOSSysSchema keyword to further limit the number of tables for which information is returned.

For more information about which system catalog tables can be used with `AlternateZOSSysSchema`, and about the function of `AlternateZOSSysSchema`, see the documentation for `APAR PK05102` by visiting:

Support for IBM mainframes

and searching for "`PK05102`".

## ArrayInputChain IBM Data Server Driver configuration keyword

Enables array input without needing pre-specified size and memory allocation requirements of normal array input.

**Equivalent CLI keyword**
ArrayInputChain

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**`db2dsdriver.cfg` configuration syntax**

```
<parameter name="ArrayInputChain" value="-1 | 0 | <positive
integer>"/>
```

**Default setting:**

Normal input array is enabled, where the array and its size must be specified before the corresponding `SQLExecute()` call is made.

**Usage notes:**

By default, array input (where an array of values is bound to an input parameter) requires the array and its size to be specified before the corresponding `SQLExecute()` function is called. An application, however, may not know the array size in advance, or the array size may be too large for the application to allocate from its pool of available memory. Under these circumstances, the application can set ArrayInputChain=-1 and use the SQL_ATTR_CHAINING_BEGIN and SQL_ATTR_CHAINING_END statement attributes to enable chaining, which allows array input without the pre-specified size and memory requirements of normal array input.

To enable chaining:

1. Set the keyword ArrayInputChain = -1.
2. Prepare and bind input parameters to the SQL statement.
3. Set the SQL_ATTR_CHAINING_BEGIN statement attribute with `SQLSetStmtAttr()`.
4. Update the bound parameters with input data and call `SQLExecute()`.
5. Repeat Step 4 for as many rows as there are in the input array.
6. Set the SQL_ATTR_CHAINING_END statement attribute with `SQLSetStmtAttr()` after the last row in the array has been processed according to Step 4.

The effect of completing these steps will be the same as if normal array input had been used.

Setting ArrayInputChain=0 (the default value) turns this array input feature off. ArrayInputChain can also be set to any positive integer which sets the array size to use for the input array.

**Restriction:** DB2 CLI does not support array input chaining for compound SQL (compiled) or compound SQL (inlined) statements.

## Authentication IBM Data Server Driver configuration keyword

Specifies the type of authentication to be used with file DSN or DSN-less connectivity.

**Equivalent CLI keyword**

Authentication

**Equivalent IBM Data Server Provider for .NET connection string keyword**

Authentication

**`db2dsdriver.cfg` configuration syntax**

```
<parameter name="Authentication" value="SERVER | SERVER_ENCRYPT |
SERVER_ENCRYPT_AES | DATA_ENCRYPT | KERBEROS | GSSPLUGIN"/>
```

**Default setting:**

`SERVER`

**Usage notes:**

This can be set in the `db2dsdriver.cfg` file for the given data source.

If `Kerberos` is specified, then the **`KRBPlugin`** can also be optionally specified. If **`KRBPlugin`** is not specified, the default plugin `IBMkrb5` will be used.

The SERVER_ENCRYPT_AES authentication type is available as of DB2 Version 9.5 Fix Pack 3.

## BigintDefaultCMapping IBM Data Server Driver configuration keyword

Specifies the default C type of BIGINT columns and parameter markers.

**Equivalent CLI keyword**
MapBigintCDefault

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="BigintDefaultCMapping" value="0 | 1 | 2"/>
```

**Default setting:**
The default C type representation for BIGINT data is SQL_C_BIGINT.

**Usage notes:**
BigintDefaultCMapping controls the C type that is used when SQL_C_DEFAULT is specified for BIGINT columns and parameter markers. Use this keyword primarily with Microsoft applications, such as Microsoft Access, which cannot handle 8-byte integers. Set BigintDefaultCMapping as follows:

- 0 - for the default SQL_C_BIGINT C type representation
- 1 - for an SQL_C_CHAR C type representation
- 2 - for an SQL_C_WCHAR C type representation

This keyword affects the behavior of CLI functions where SQL_C_DEFAULT might be specified as a C type, such as SQLBindParameter(), SQLBindCol(), and SQLGetData()

## BiDiCCSID IBM Data Server Driver configuration keyword

Overrides the Bidirectional (BIDI) coded character set identifiers (CCSID).

**Equivalent CLI keyword**
BIDI

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="BiDiCCSID" value="ccsid"/>
```

**Default setting:**

**Usage notes:**
Specifies to perform bidirectional transformation on data, and overrides the BIDI CCSID.

## ClientAccountingString IBM Data Server Driver configuration keyword

2 Sets the client accounting string that is sent to a database.

**Equivalent CLI keyword**
ClientAcctStr

**Equivalent IBM Data Server Provider for .NET connection string keyword**
ClientAccountingString

**db2dsdriver.cfg configuration syntax**
```
<parameter name="ClientAccountingString" value="accounting string"/>
```

**Default setting:**
None

**Equivalent environment or connection attribute:**
SQL_ATTR_INFO_ACCTSTR

**Usage notes:**

This option allows an application to set the client accounting string that is sent to the database. Applications that do not offer the accounting string by default can take advantage of this keyword to provide this information.

Note the following conditions:

- When the value is being set, some servers might not handle the entire length provided and might truncate the value.
- DB2 for z/OS and OS/390® servers support up to a length of 200 characters.
- To ensure that the data is converted correctly when transmitted to a host system, use only the characters A to Z, 0 to 9, and the underscore (_) or period (.).

## ClientApplicationName IBM Data Server Driver configuration keyword

Sets the client application name that is sent to a database.

**Equivalent CLI keyword**
ClientApplName

**lslsEquivalent IBM Data Server Provider for .NET connection string keyword**
ClientApplicationName

**db2dsdriver.cfg configuration syntax**
```
<parameter name="ClientApplicationName" value="application name"/>
```

**Default setting:**
None

**Equivalent environment or connection attribute:**
SQL_ATTR_INFO_APPLNAME

**Usage notes:**

This option allows an application to set the client application name that is sent to the database. Applications that do not offer the application name by default can take advantage of this keyword to provide this information.

Note the following conditions:

- When the value is being set, some servers might not handle the entire length provided and might truncate the value.
- DB2 for z/OS and OS/390 servers support up to a length of 32 characters.
- To ensure that the data is converted correctly when transmitted to a host system, use only the characters A to Z, 0 to 9, and the underscore (_) or period (.).

## ClientUserID IBM Data Server Driver configuration keyword

Sets the client user ID that is sent to a database.

**Equivalent CLI keyword**
ClientUserID

**Equivalent IBM Data Server Provider for .NET connection string keyword**
ClientUserIDClientUserID

**db2dsdriver.cfg configuration syntax**
```
<parameter name="ClientUserID" value="userid"/>
```

**Default setting:**
None

**Equivalent environment or connection attribute:**
SQL_ATTR_INFO_USERID

**Usage notes:**

This option allows an application to set the client user ID (accounting user ID) that is sent to the database. Applications that do not offer the user ID by default can take advantage of this keyword to provide this information.

Note the following conditions:

- When the value is being set, some servers might not handle the entire length provided and might truncate the value.
- DB2 for z/OS and OS/390 servers support up to a length of 16 characters.
- This user ID is not to be confused with the authentication user ID. This user ID is for identification purposes only and is not used for any authorization.
- To ensure that the data is converted correctly when transmitted to a host system, use only the characters A to Z, 0 to 9, and the underscore (_) or period (.).

## ClientWorkstationName IBM Data Server Driver configuration keyword

Sets the client workstation name that is sent to a database.

**Equivalent CLI keyword**
ClientWrkStnName

**Equivalent IBM Data Server Provider for .NET connection string keyword**
ClientWorkstationName

**db2dsdriver.cfg configuration syntax**
```
<parameter name="ClientWorkstationName" value="workstation name"/>
```

**Default setting:**
None

**Equivalent environment or connection attribute:**
SQL_ATTR_INFO_WRKSTNNAME

**Usage notes:**

This option allows an application to set the client workstation name that is sent to the database. Applications that do not offer the client workstation name by default can take advantage of this keyword to provide this information.

Note the following conditions:

- When the value is being set, some servers might not handle the entire length provided and might truncate the value.
- DB2 for z/OS and OS/390 servers support up to a length of 18 characters.
- To ensure that the data is converted correctly when transmitted to a host system, use only the characters A to Z, 0 to 9, and the underscore (_) or period (.).

## CLIPatch1 IBM Data Server Driver configuration keyword

Specifies a work-around for known CLI/ODBC application problems.

**Equivalent CLI keyword**
Patch1

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="CLIPatch1" value="{ 0 | 1 | 2 | 4 | 8 | 16 | ...
}"/>
```

**Default setting:**
Use no work-arounds.

**Usage notes:**
This keyword is used to specify a work-around for known problems with ODBC applications. The value specified can be for none, one, or multiple work-arounds. The patch values specified here are used in conjunction with any CLIPatch2 values that might also be set.

Using the DB2 CLI/ODBC Settings notebook you can select one or more patches to use. If you set the values in the `db2cli.ini` file itself and want to use multiple patch values then simply add the values together to form the keyword value. For example, if you want the patches 1, 4, and 8, then specify CLIPatch1=13.

- 0 = No work around (default)

*Table 2-1. CLIPatch1 CLI/ODBC configuration keyword values*

| Value | Description |
|---|---|
| 4 | Changes input timestamp data to date data if the time and fraction part of the timestamp are zero. For example, {ts 'YYYY-MM-DD 00:00:00'} is changed to {d 'YYYY-MM-DD'}. This value is typically needed for older versions of Microsoft Access. |
| 8 | Changes input timestamp data to time data if the date part of the timestamp is either 1899-12-30 or 1900-01-01. For example, {ts '1899-12-30 HH:MM:SS'} is changed to {t 'HH:MM:SS'}. This value is typically needed for older versions of Microsoft Access. |

*Table 2-1. CLIPatch1 CLI/ODBC configuration keyword values (continued)*

| Value | Description |
|---|---|
| 64 | Null-terminates output GRAPHIC strings. This value is typically needed by Microsoft Access in a double-byte (DBCS) environment. |
| 128 | Disables the default performance optimization behavior for the MSysConf table associated with some Microsoft applications.<br><br>Microsoft applications, such as Microsoft Access, use a configuration table called MSysConf. Once these applications successfully connect to a database, they will typically issue the following query: "SELECT Config, nValue FROM MSysConf". Because the MSysConf table does not exist in a DB2 database by default, this query fails with the error "SQL0204N "MSysConf" is an undefined name.". Microsoft applications can handle this error and continue processing, however, issuing the query across the network to the DB2 server incurs overhead.<br><br>To enhance performance, DB2 CLI assumes that this query will always fail, so when it detects that an application is trying to execute this query, it automatically returns an error with an SQLSTATE of S0002 (Table not found). The query, therefore, is never sent to the server. If, however, the user has created the MSysConf configuration table in the database and wants the application to access it, this CLIPatch1 value can be set to disable the performance optimization and allow the query to be executed. |
| 256 | Service use only |
| 512 | Service use only |
| 1024 | Returns SQL_SUCCESS_WITH_INFO instead of SQL_NO_DATA_FOUND from the SQLExecute() and SQLExecDirect() functions if the executed UPDATE or DELETE statement affected no rows. This value might be needed by some Microsoft Visual Basic applications. |
| 4096 | Prevents a COMMIT from being issued after closing a cursor in autocommit mode. |
| 8192 | Returns an extra result set after invoking a stored procedure. This extra result set has one row and consists of the output values of the stored procedure. This CLIPatch1 value might be needed by some Powerbuilder applications that require an extra result set. |
| 32768 | Forces the driver to make Microsoft Query applications work with DB2 MVS™ synonyms. |

*Table 2-1. CLIPatch1 CLI/ODBC configuration keyword values (continued)*

| Value | Description |
|---|---|
| 65536 | Deprecated |
| 131072 | Deprecated |
| 262144 | Deprecated |
| 524288 | Deprecated |
| 1048576 | Service use only |
| 2097152 | Service use only |

## CLIPatch2 IBM Data Server Driver configuration keyword

Specifies a work-around for known CLI and ODBC application problems.

**Equivalent CLI keyword**
Patch2

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="CLIPatch2" value="patch value 1, patch value 2,
patch value 3, ..."/>
```

**Default setting:**
Use no work-arounds

**Usage notes:**
This keyword is used to specify a work-around for known problems with CLI and ODBC applications. The value specified can be for none, one, or multiple work-arounds. The patch values specified here may be used in conjunction with any **CLIPatch1** values that may also be set.

When specifying multiple patches, the values are specified in a comma delimited string (unlike the **CLIPatch1** option where the values are added together and the sum is used).
- `0` = No work around (default)

To set **CLIPatch2** values `3`, `4` and `8` you would specify:
```
CLIPatch2="3, 4, 8"
```

*Table 2-2. CLIPatch2 CLI/ODBC configuration keyword values*

| Value | Description |
|---|---|
| 1 | Deprecated |
| 3 | Service use only |
| 4 | Deprecated |
| 5 | Deprecated |
| 6 | Forces the driver to return a message indicating that scrollable cursors are not supported. This setting is needed by some applications (such as Visual Basic) that make use of LOBs or that do not need or want scrollable cursors to be used, even though they have been explicitly requested by the application. |

*Table 2-2. CLIPatch2 CLI/ODBC configuration keyword values (continued)*

| Value | Description |
|---|---|
| 7 | Maps all GRAPHIC column data types to the CHAR column data type. The precision of a GRAPHIC column will also be doubled; for example, GRAPHIC(20) will be reported as CHAR(40). |
| 8 | Ignores catalog search arguments in schema calls. |
| 11 | SQLGetInfo() reports that catalog names are supported for Visual Basic stored procedures. |
| 12 | Deprecated |
| 13 | Prevents keywords in the `db2cli.ini` initialization file from being appended to the output connection string. |
| 14 | Deprecated |
| 15 | Causes a period separator to be used instead of the default locale's decimal separator in character output. |
| 16 | Deprecated |
| 17 | Deprecated |
| 18 | Attempts to replace literals with parameter markers for inefficient applications that use literals repeatedly. It is only applicable to INSERT SQL statements with the VALUES clause using only literals. Coding your application properly to use parameter markers is the best solution. |
| 19 | Removes parentheses from the ON clause of an outer join, where the outer join is an ODBC escape sequence and the server is DB2 for MVS Version 5. DB2 for MVS Version 5 does not currently support the ODBC syntax where parentheses are permitted in the ON clause of an outer join clause. Setting this **CLIPatch2** value allows the outer join escape sequence to be used against DB2 for MVS Version 5. This value should only be set when the server is DB2 for MVS Version 5. |
| 20 | Forces the driver to rewrite the BETWEEN predicate when the server is DB2 for MVS. DB2 for MVS does not currently support the BETWEEN predicate with parameter markers as both operands. Setting this **CLIPatch2** value causes (expression ? BETWEEN ?) to be rewritten as (expression >= ? and expression <= ?). |
| 21 | Deprecated |

*Table 2-2. CLIPatch2 CLI/ODBC configuration keyword values (continued)*

| Value | Description |
|---|---|
| 22 | Causes SQLGetInfo() to report SQL_OUTER_JOINS=NO and SQL_OJ_CAPABILITIES=0. This prevents the application from using outer joins where they are not supported, thus ensuring that the outer join queries do not fail. |
| 23 | Deprecated |
| 24 | Reports TIME data as SQL_CHAR data. This patch value is used as a workaround for Microsoft Access applications. |
| 25 | Removes trailing zeros in the CHAR representation of DECIMAL columns; used as a workaround for Microsoft Access applications. |
| 28 | Deprecated |
| 29 | Removes leading zeroes in the string representation of DECIMAL values x, where 1 > x > -1; used as a workaround for ADO applications with some MDAC versions. |
| 30 | Disables stored procedure caching optimization. |
| 31 | Deprecated |
| 32 | Deprecated |
| 33 | Returns the ISO version of timestamp data when converted to CHAR, rather than the ODBC version. |
| 34 | Deprecated |
| 38 | Turns statement caching off |
| 42 | Prevents the FOR UPDATE clause from being used with keyset cursors. By default, most applications expect keyset cursors to be updatable, however, if this is not required, then this CLIPatch2 value makes the cursor read-only (but still scrollable and sensitive to changes made by others). |
| 50 | Frees LOB locators when SQLFetch() is executed, rather than when a COMMIT is issued. This **CLIPatch2** value frees the locators that are used internally when applications fetch LOB data without binding the LOB columns with SQLBindCol() (or equivalent descriptor APIs). Locators that are explicitly returned to the application must still be freed by the application. This **CLIPatch2** value can be used to avoid scenarios where an application receives SQLCODE = -429 (no more locators). |
| 56 | Allows client support for Early Close Cursors for those servers that do not support it as in the case of DB2 UDB for OS/390 version 7 or earlier. |

*Table 2-2. CLIPatch2 CLI/ODBC configuration keyword values (continued)*

| Value | Description |
|---|---|
| 57 | Allows calling a stored procedure that returns a NULL output parameter value without providing an output indicator pointer. This is normally applicable to older versions of Borland Delphi products. |
| 58 | DateTime values inserted into the database that cause truncation errors can be downgraded to a truncation warning using this **CLIPatch2** value. |
| 61 | When data is given to the client from an SQL_CHAR there may be right padded spaces. This patch value strips off right padded single byte spaces, but not double byte spaces. This behavior partially mimics the Neon Shadow Driver behavior |
| 66 | Allows applications to retrieve the regional setting that affects decimal separators in a Windows environment. The regional setting is normally ignored by default. This patch value is ignored if **CLIPatch2**=15 or **db2set** registry variables **DB2COUNTRY**, **DB2TERRITORY**, or **DB2CODEPAGE** are set. The only supported decimal separators are the period and comma. |
| 78 | Alters the behavior of the SQLGetPosition() function when the source LOB value is in a DBCLOB column on DB2 for z/OS and OS/390 Version 7.1 or later. This **CLIPatch2** value is available in DB2 Version 9.5 Fix Pack 2 and later and it causes the SQLGetPosition() function to query SYSIBM.SYSDUMMYU instead of SYSIBM.SYSDUMMY1. |

## CommProtocol IBM Data Server Driver configuration keyword

Communication protocol used to connect to the data source.

**Equivalent CLI keyword**
Protocol

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="CommProtocol" value="TCPIP |
TCPIP6 |                TCPIP4 |                IPC |
                LOCAL"/>
```

**Default setting:**
None.

**Usage notes:**
This can be set in the db2dsdriver.cfg file for the given data source. There

is no default setting. However, this is a required parameter. If CommProtocol is not specified at one of the levels (alias, database or global) the connection will fail.

TCP/IP is the only protocol supported when using a File DSN. Set the option to the string TCPIP (without the slash).

IPC connectivity can be specified by setting CommProtocol to either **IPC** or **LOCAL**.

When CommProtocol = **IPC** | **LOCAL** the IPCInstance keyword must also be set.

## ConcurrentAccessResolution IBM Data Server Driver configuration keyword

Specifies the concurrent access resolution to use.

**Equivalent CLI keyword**

ConcurrentAccessResolution

**Equivalent IBM Data Server Provider for .NET connection string keyword**

ConcurrentAccessResolution

**db2dsdriver.cfg configuration syntax**

```
<parameter name="ConcurrentAccessResolution" value="0 |
CurrentlyCommitted | WaitForOutcome | SkipLockedData"/>
```

**Default setting:**

DB2 CLI does not supply a prepare option, and the currently committed behavior is determined by the database configuration.

**Usage notes:**

This keyword specifies a prepare attribute that overrides the default behavior specified for cursor stability (CS) scans.

- 0 = No setting. The client does not supply a prepare option.
- CurrentlyCommitted = Use currently committed semantics. DB2 CLI flows "currently committed" on every prepare, which means that the database manager can use the currently committed version of the data for applicable scans when the data is in the process of being updated or deleted. Rows in the process of being inserted can be skipped. This setting applies when the isolation level in effect is Cursor Stability or Read Stability (for Read Stability it skips uncommited inserts only) and is ignored otherwise. Applicable scans include read-only scans that can be part of a read-only statement as well as a non read-only statement. The settings for the registry variables **DB2_EVALUNCOMMITTED**, **DB2_SKIPDELETED**, and **DB2_SKIPINSERTED** do not apply to scans using currently committed. However, the settings for these registry variables still apply to scans that do not use currently committed.
- WaitForOutcome= Wait for outcome. DB2 CLI flows "wait for outcome" on every prepare, which means that Cursor Stability and higher scans wait for the commit or rollback when encountering data in the process of being updated or deleted. Rows in the process of being inserted are not skipped. The settings for the registry variables DB2_EVALUNCOMMITTED, DB2_SKIPDELETED, and DB2_SKIPINSERTED no longer apply.
- SkipLockedData = Skip locked data. DB2 CLI flows "skip locked data" on every prepare, which means that currently committed semantics are

used and rows in the process of being inserted are skipped. This option is not supported on DB2 Database for Linux, UNIX, and Windows. If specified, this setting is ignored.

For DB2 Database for Linux, UNIX, and Windows, use this keyword to override the default behavior for currently committed that is defined by the **cur_commit** configuration parameter. For DB2 for z/OS, use this keyword to enable currently committed behavior. There is no equivalent database configuration parameter available on DB2 for z/OS for specifying this behavior.

DB2 z/OS Version 10 only supports INSERT and DELETE operations of currently committed.

## ConnectionLevelLoadBalancing IBM Data Server Driver configuration keyword

Enables DB2 Connect DisableSysplex support for a particular database.

**Equivalent CLI keyword**
DisableSysplex

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="ConnectionLevelLoadBalancing" value=""/>
```

**Default setting:**
N/A

**Usage notes:**
Explicitly enable/disable DB2 Connection Level Load balancing support for a particular database.

## ConnectionLifetimeInPool IBM Data Server Driver configuration keyword

Sets the number of seconds a connection remains idle in the connection pool.

**Equivalent CLI keyword**
N/A

**Equivalent IBM Data Server Provider for .NET connection string keyword**
Connection Lifetime

**db2dsdriver.cfg configuration syntax**
```
<parameter name="ConnectionLifetimeInPool" value="60"/>
```

**Default setting:**
60

**Usage notes:**
The amount of time (in seconds) that the connection can remain idle in the connection pool.

## ConnectionTimeout IBM Data Server Driver configuration keyword

Specifies the time in seconds to wait for a reply when trying to establish a connection to a server before terminating the attempt and generating a communication timeout.

**Equivalent CLI keyword**
ConnectTimeout

**Equivalent IBM Data Server Provider for .NET connection string keyword**
Timeout

**db2dsdriver.cfg configuration syntax**
```
<parameter name="ConnectionTimeout" value="0 | 1 |2 | ... | 32767"/>
```

**Default setting:**
The client waits indefinitely for a reply from the server when trying to establish a connection.

**Equivalent connection attribute:**
SQL_ATTR_LOGIN_TIMEOUT

**Usage notes:**

If **ConnectionTimeout** is set and client reroute is enabled, a connection will be attempted only once to the original server and once to the alternative server. Since the **ConnectionTimeout** value is used when attempting to connect to each server, the maximum waiting time will be approximately double the specified value for **ConnectionTimeout**. If neither server can be reached within the amount of time specified by the keyword, the following error message will be received:

```
SQL30081N  A communication error has been detected.  Communication
   protocol being used: "TCP/IP". Communication API being used:
   "SOCKETS".  Location where the error was detected: "<ip address>".
   Communication function detecting the error: "<failing function>".
   Protocol specific error code(s): "<error code>", "*", "*".
   SQLSTATE=08001
```

If **ConnectionTimeout** is set and Sysplex exploitation is enabled, a connection will be attempted only once for each of the Sysplex members. Since the **ConnectionTimeout** value is used when attempting to connect to each Sysplex member, the maximum waiting time will be approximately equal to the number of Sysplex members, times the amount of time specified by the **ConnectionTimeout** keyword.

If both **tcpipConnectTimeout** and **ConnectionTimeout** values are set, then the **tcpipConnectTimeout** keyword value should be less than the **ConnectionTimeout** keyword value.
**ConnectionTimeout** only applies to the TCPIP protocol and is not supported for connections to databases cataloged on a SOCKS-enabled TCP/IP node.

## ConnectNodeNumber IBM Data Server Driver configuration keyword

Specifies the database partition server to which a connection is to be made.

**Equivalent CLI keyword**
ConnectNode

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="ConnectNodeNumber" value="integer value from 0 to
999 |  SQL_CONN_CATALOG_NODE"/>
```

**Default setting:**
Database partition server which is defined with port 0 on the machine is used.

**Equivalent connection attribute:**
SQL_ATTR_CONNECT_NODE

**Usage notes:**

Used to specify the target database partition server that you want to connect to. This keyword (or attribute setting) overrides the value of the environment variable DB2NODE. Can be set to:

- an integer between 0 and 999
- SQL_CONN_CATALOG_NODE

If this variable is not set, the target defaults to the database partition server that is defined with port 0 on the machine.

**Note:** This keyword does not affect the Control Center. The Control Center always connects to the catalog partition referred to by the SQL_CONN_CATALOG_NODE setting.

## CurrentFunctionPath IBM Data Server Driver configuration keyword

Specifies the schema used to resolve function references and data type references in dynamic SQL statements.

**Equivalent CLI keyword**
CurrentFunctionPath

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**`db2dsdriver.cfg` configuration syntax**

```
<parameter name="CurrentFunctionPath"
value="current_function_path"/>
```

**Default setting:**
See description below.

**Usage notes:**

This keyword defines the path used to resolve function references and data type references that are used in dynamic SQL statements. It contains a list of one or more schema-names, where schema-names are enclosed in double quotation marks and separated by commas.

The default value is "SYSIBM","SYSFUN",X where X is the value of the USER special register delimited by double quotation marks. The schema SYSIBM does not need to be specified. If it is not included in the function path, then it is implicitly assumed as the first schema.

This keyword is used as part of the process for resolving unqualified function and stored procedure references that may have been defined in a schema name other than the current user's schema. The order of the schema names determines the order in which the function and procedure names will be resolved.

## CurrentPackagePath IBM Data Server Driver configuration keyword

Issues `SET CURRENT PACKAGE PATH = schema1, schema2, ...` after every connection.

**Equivalent CLI keyword**
CurrentPackagePath

**Equivalent IBM Data Server Provider for .NET connection string keyword**
Not available.

**db2dsdriver.cfg configuration syntax**
```
<parameter name="CurrentPackagePath" value="schema1, schema2, ..."/>
```

**Default setting:**
The clause is not appended.

**Equivalent connection attribute:**
SQL_ATTR_CURRENT_PACKAGE_PATH

**Usage notes:**

When set, this option issues the statement `SET CURRENT PACKAGE PATH = schema1, schema2, ...` after every connection to the database. This setting specifies the list of schema names (collection identifiers) that will be searched when there is a package from a different schema.

This keyword is best suited for use with ODBC static processing applications.

## CurrentPackageSet IBM Data Server Driver configuration keyword

Issues the SET CURRENT PACKAGESET statement after every connection.

**Equivalent CLI keyword**
CurrentPackageSet

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="CurrentPackageSet" value="schema name"/>
```

**Default setting:**
The clause is not appended.

**Equivalent connection attribute:**
SQL_ATTR_CURRENT_PACKAGE_SET

**Usage notes:**

This option issues the SET CURRENT PACKAGESET SQL statement with the CurrentPackageSet value after every connection to a database. By default this clause is not appended.

The SET CURRENT PACKAGESET SQL statement sets the schema name (collection identifier) that is used to select the package to use for subsequent SQL statements.

Applications issue dynamic SQL statements. Using this option you can control the privileges used to run these statements:

- Choose a schema to use when running SQL statements from applications.

- Ensure the objects in the schema have the required privileges and then rebind accordingly.
- Set the CurrentPackageSet option to this schema.

The SQL statements from the applications will now run under the specified schema and use the privileges defined there.

The following package set names are reserved: NULLID, NULLIDR1, NULLIDRA.

If both the Reopt and CurrentPackageSet keywords are specified, CurrentPackageSet takes precedence.

## CurrentRefreshAge IBM Data Server Driver configuration keyword

Sets the value of the CURRENT REFRESH AGE special register.

**Equivalent CLI keyword**
CurrentRefreshAge

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="CurrentRefreshAge" value="0 | ANY | positive
integer"/>
```

**Default setting:**
Only materialized query tables defined with REFRESH IMMEDIATE can be used to optimize the processing of a query.

**Usage notes:**

Setting this keyword sets the value of the CURRENT REFRESH AGE special register.

## CurrentSchema IBM Data Server Driver configuration keyword

Specifies the schema used in a SET CURRENT SCHEMA statement upon a successful connection.

**Equivalent CLI keyword**
CurrentSchema

**Equivalent IBM Data Server Provider for .NET connection string keyword**
CurrentSchema

**db2dsdriver.cfg configuration syntax**

```
<parameter name="CurrentSchema" value="schema name"/>
```

**Default setting:**
No statement is issued.

**Usage notes:**

Upon a successful connect, if this option is set, a SET CURRENT SCHEMA statement is sent to the DBMS. This allows the end user or application to name SQL objects without having to qualify them by schema name.

## CurrentSQLID IBM Data Server Driver configuration keyword

Specifies the ID used in a SET CURRENT SQLID statement sent to the DBMS upon a successful connection.

**Equivalent CLI keyword**
CurrentSQLID

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="CurrentSQLID" value="current_sqlid"/>
```

**Default setting:**
No statement is issued.

**Usage notes:**

Upon a successful connection, if this option is set, a SET CURRENT SQLID statement is sent to the DBMS. This allows the end user and the application to name SQL objects without having to qualify them by schema name.

## DateDefaultCMapping IBM Data Server Driver configuration keyword

Specifies the default C type of DATE columns and parameter markers.

**Equivalent CLI keyword**
MapDateCDefault

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="DateDefaultCMapping" value="0 | 1 | 2"/>
```

**Default setting:**
The default C type representation for DATE data is SQL_C_TYPE_DATE.

**Usage notes:**

DateDefaultCMapping controls the C type that is used when SQL_C_DEFAULT is specified for DATE columns and parameter markers. This keyword should be used primarily with Microsoft applications, such as Microsoft Access, which assume SQL_C_CHAR as the default C type for datetime values. Set DateDefaultCMapping as follows:

- 0 - for the default SQL_C_TYPE_DATE C type representation: a struct containing numeric members for year, month and day
- 1 - for an SQL_C_CHAR C type representation: "2009-01-01"
- 2 - for an SQL_C_WCHAR C type representation: "2009-01-01" in UTF-16.

This keyword affects the behavior of CLI functions where SQL_C_DEFAULT can be specified as a C type, such as `SQLBindParameter()`, `SQLBindCol()`, and `SQLGetData()`.

## DateDefaultDescribeMapping IBM Data Server Driver configuration keyword

Controls the SQL data type returned when DATE columns and parameter markers are described.

**Equivalent CLI keyword**
MapDateDescribe

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="DateDefaultDescribeMapping" value="0 | 1 | 2"/>
```

**Default setting:**
The default SQL data type for DATE data is returned: SQL_DATE for ODBC 2.0 or SQL_TYPE_DATE for ODBC 3.0.

**Usage notes:**

To control the SQL data type that is returned when DATE columns and parameter markers are described, set DateDefaultDescribeMapping as follows:

- 0 - to return the default SQL data type: SQL_DATE for ODBC 2.0 or SQL_TYPE_DATE for ODBC 3.0
- 1 - to return the SQL_CHAR SQL data type
- 2 - to return the SQL_WCHAR SQL data type

Only the following DB2 CLI functions are affected by setting DateDefaultDescribeMapping:

- `SQLColumns()`
- `SQLDescribeCol()`
- `SQLDescribeParam()`
- `SQLGetDescField()`
- `SQLGetDescRec()`
- `SQLProcedureColumns()`
- `SQLSpecialColumns()`

## DateTimeStringFormat IBM Data Server Driver configuration keyword

Specifies the format to use when inserting date or time data into character columns.

**Equivalent CLI keyword**
DateTimeStringFormat

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="DateTimeStringFormat" value="JIS | ISO | EUR |
USA"/>
```

**Default setting:**
The JIS format is used when date or time data is inserted into character columns.

**Usage notes:**

The DateTimeStringFormat keyword controls the format in which date or time data is inserted into character columns. This setting affects the insertion of SQL_C_TYPE_DATE, SQL_C_TYPE_TIME, or SQL_C_TYPE_TIMESTAMP, or SQL_C_TYPE_TIMESTAMP_EXT data into the following column types:

- SQL_CHAR
- SQL_VARCHAR

- SQL_LONGVARCHAR
- SQL_CLOB

This keyword also affects the format of date or time columns that are retrieved into character strings. For example, retrieving data from an SQL_TYPE_TIMESTAMP column into an SQL_C_CHAR string will be affected by the setting of this keyword.

The four setting values are as follows:

| Format | Date | Time | Timestamp |
|---|---|---|---|
| JIS | yyyy-mm-dd | hh:mm:ss | yyyy-mm-dd hh:mm:ss.ffffff |
| ISO | yyyy-mm-dd | hh.mm.ss | yyyy-mm-dd-hh.mm.ss.ffffff |
| EUR | dd.mm.yyyy | hh.mm.ss | yyyy-mm-dd hh:mm:ss.ffffff* |
| USA | mm/dd/yyyy | hh:mm AM or PM | yyyy-mm-dd hh:mm:ss.ffffff* |
| *Timestamps will take the default format if EUR or USA is specified. The default format is JIS. | | | |

## DecimalFloatDefaultDescribeMapping IBM Data Server Driver configuration keyword

Specifies the default C type and reported data type of DECFLOAT columns and parameter markers.

**Equivalent CLI keyword**
MapDecimalFloatDescribe

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="DecimalFloatDefaultDescribeMapping" value="0 | 1|
2| 3"/>
```

**Default setting:**
0

**Usage notes:**

DecimalFloatDefaultDescribeMapping controls the default C type to be used for columns and parameters with a data type of DECFLOAT. It affects the behavior of functions for which SQL_C_DEFAULT can be specified as the C type of a column or parameter. Examples of such functions include `SQLBindParameter()`, `SQLBindCol()`, and `SQLGetData()`,

DecimalFloatDefaultDescribeMapping also controls the type that will be reported for columns and parameters that have a data type of DECFLOAT. This affects CLI functions that return information about parameters and columns. Examples of such functions include `SQLColAttribute()` and `SQLDescribeParam()`.

Use this configuration keyword for applications that cannot handle decimal float types or when you would rather always deal with decimal float types as some other type.

Here are the allowed values:

*Table 2-3. Allowed values*

| Value | DECFLOAT columns and parameters are reported as being this type | DECFLOAT columns and parameters use this default C type |
|---|---|---|
| 0 | SQL_DECFLOAT | SQL_C_CHAR |
| 1 | SQL_VARCHAR | SQL_C_CHAR |
| 2 | SQL_WVARCHAR | SQL_C_WCHAR |
| 3 | SQL_DOUBLE | SQL_C_DOUBLE |

## DecimalFloatRoundingMode IBM Data Server Driver configuration keyword

Sets the rounding mode when working with servers that support the DECFLOAT SQL type.

**Equivalent CLI keyword**
DecimalFloatRoundingMode

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="DecimalFloatRoundingMode" value="0 | 1 | 2 | 3 |
4"/>
```

**Default setting:**
0 (Half even rounding mode)

**Equivalent connection attribute:**
SQL_ATTR_DECFLOAT_ROUNDING_MODE

**Usage notes:**

The decimal float rounding mode determines what type of rounding will be used if a value is put into a DECFLOAT variable or column but the value has more digits than are allowed in the DECFLOAT data type. This can occur when inserting, updating, selecting, converting from another type, or as the result of a mathematical operation.

The value of SQL_ATTR_DECFLOAT_ROUNDING_MODE determines the decimal float rounding mode that will be used for new connections unless another mode is specified by a connection attribute for that connection. For any given connection both DB2 will use the same decimal float rounding mode for all action initiated as part of that connection.

When your applications are connecting to a DB2 Database for Linux, UNIX, and Windows Version 9.5 server, you must set the decimal float rounding mode on the database client to the same mode that is set on the server. If you set the decimal float rounding mode on the client to a value that is different from the decimal float rounding mode that is set on the database server, the database server will return SQL0713N on connection.

The settings correspond to these decimal float rounding modes:

- 0 = Half even (default)
- 1 = Half up

- 2 = Down
- 3 = Ceiling
- 4 = Floor

The different modes are:

**Half even (default)**
In this mode DB2 uses the number that will fit in the target variable and that is closest to the original value. If two numbers are equally close, they use the one that is even. This mode produces the smallest rounding errors over large amounts of data.

**Half up**
In this mode DB2 uses the number that will fit in the target variable and that is closest to the original value. If two numbers are equally close, they use the one that is greater than the original value.

**Down** In this mode DB2 uses the number that will fit in the target variable and that is closest to the original value and for which the absolute value is not greater than the absolute value of the original value. You can also think of this as rounding toward zero or as using ceiling for negative values and using floor for positive values.

**Ceiling**
In this mode DB2 uses the smallest number that will fit in the target variable and that is greater than or equal to the original value.

**Floor** In this mode DB2 uses the largest number that will fit in the target variable and that is less than or equal to the original value.

## DisableAliasesInMetadata IBM Data Server Driver configuration keyword

Controls whether the CLI schema APIs report aliases in the metadata results.

**Equivalent CLI keyword**
ReturnAliases

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="DisableAliasesInMetadata" value="0 | 1"/>
```

**Default setting:**
There is no default setting

**Usage notes:**

This keyword disables the consideration of aliases (or synonyms) when qualifying rows for metadata procedures. Not considering aliases can provide significant performance benefits by avoided costly joins with the base tables to determine the addition tables that qualify for a given query.

- 0 : Aliases will be considered when qualifying rows for metadata procedures.
- 1 : Aliases will not be considered when qualifying rows for metadata procedures (better performance.)

The following CLI APIs are affected by this keyword :

- SQLColumns()
- SQLColumnPrivileges()
- SQLTables()
- SQLTablePrivileges()
- SQLStatistics()
- SQLSpecialColumns()
- SQLForeignKeys()
- SQLPrimaryKeys()

## DisableAsyncQueryExecution IBM Data Server Driver configuration keyword

Disables the ability to execute queries asynchronously.

**Equivalent CLI keyword**
AsyncEnable

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="DisableAsyncQueryExecution" value="0 | 1"/>
```

**Default setting:**
Queries can be executed asynchronously.

**Usage notes:**
This option allows you to disable support that allows queries to execute asynchronously. This only benefits applications that were written to take advantage of this feature by setting the SQL_ATTR_ASYNC_ENABLE attribute using SQLSetStmtAttr() or SQLSetConnectAttr().

- 0 = Allow queries to be executed asynchronously. The application must also enable the asynchronous function by setting SQL_ATTR_ASYNC_ENABLE using SQLSetStmtAttr() or SQLSetConnectAttr(). (default)
- 1 = Queries are not executed asynchronously

Once a function has been called asynchronously, only the original function, SQLAllocHandle(), SQLCancel(), SQLSetStmtAttr(), SQLGetDiagField(), SQLGetDiagRec(), or SQLGetFunctions() can be called on the statement handle, until the original function returns a code other than SQL_STILL_EXECUTING. Any other function called on any other statement handle under the same connection returns SQL_ERROR with an SQLSTATE of HY010 (Function sequence error).

## DisableAutoCommit IBM Data Server Driver configuration keyword

Disables the application's autocommit after each statement.

**Equivalent CLI keyword**
AutoCommit

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="DisableAutoCommit" value="0 | 1"/>
```

**Default setting:**
Each statement is treated as a single, complete transaction.

**Equivalent connection attribute:**
SQL_ATTR_AUTOCOMMIT

**Usage notes:**

To be consistent with ODBC, DB2 CLI defaults with DisableAutoCommit on, which means each statement is treated as a single, complete transaction. This keyword can provide an alternative default, but will only be used if the application does not specify a value for SQL_ATTR_AUTOCOMMIT.

- 0 = SQL_ATTR_AUTOCOMMIT_ON (default)
- 1 = SQL_ATTR_AUTOCOMMIT_OFF

**Note:** Most ODBC applications assume the default of DisableAutoCommit to be on. Extreme care must be used when overriding this default during runtime as the application can depend on this default to operate properly.

This keyword also allows you to specify whether autocommit should be enabled in a Distributed Unit of Work (DUOW) environment. If a connection is part of a coordinated Distributed Unit of Work, and DisableAutoCommit is not set, the default does not apply; implicit commits arising from autocommit processing are suppressed. If DisableAutoCommit is set to 0, and the connection is part of a coordinated Distributed Unit of Work, the implicit commits are processed. This can result in severe performance degradation, and possibly other unexpected results elsewhere in the DUOW system. However, some applications might not work at all unless this is enabled.

A thorough understanding of the transaction processing of an application is necessary, especially applications written by a third party, before applying it to a DUOW environment.

## DisableBinaryDataSupport IBM Data Server Driver configuration keyword

Specifies whether binary data types are reported as binary or character data types.

**Equivalent CLI keyword**
BitData

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**`db2dsdriver.cfg` configuration syntax**

```
<parameter name="DisableBinaryDataSupport" value="0 | 1"/>
```

**Default setting:**
Report `FOR BIT DATA` and BLOB data types as binary data types.

**Usage notes:**

This option allows you to specify whether ODBC binary data types (SQL_BINARY, SQL_VARBINARY, SQL_LONGVARBINARY, and SQL_BLOB), are reported as binary type data. IBM DBMSs support columns with binary data types by defining CHAR, VARCHAR, and LONG VARCHAR columns with the `FOR BIT DATA` attribute. DB2 Database for Linux, UNIX, and Windows will also support binary data via the BLOB data type (in this case it is mapped to a CLOB data type).

Only set DisableBinaryDataSupport = 1 if you are sure that all columns defined as FOR BIT DATA or BLOB contain only character data, and the application is incapable of displaying binary data columns.
- 0 = report FOR BIT DATA and BLOB data types as binary data types (default).
- 1 = report FOR BIT DATA and BLOB data types as character data types.

## DisableCursorHold IBM Data Server Driver configuration keyword

Controls the effect of a transaction completion on open cursors.

**Equivalent CLI keyword**
CursorHold

**Equivalent IBM Data Server Provider for .NET connection string keyword**
DisableCursorHold

**db2dsdriver.cfg configuration syntax**
```
<parameter name="DisableCursorHold" value="0 | 1"/>
```

**Default setting:**
Cursors are not destroyed.

**Equivalent statement attribute:**
SQL_ATTR_CURSOR_HOLD

**Usage notes:**

This option controls the effect of a transaction completion on open cursors.
- 0 = SQL_CURSOR_HOLD_ON, the cursors are not destroyed when the transaction is committed (default).
- 1 = SQL_CURSOR_HOLD_OFF, the cursors are destroyed when the transaction is committed.

**Note:** Cursors are always closed when transactions are rolled back.

This option affects the result returned by SQLGetInfo() when called with SQL_CURSOR_COMMIT_BEHAVIOR or SQL_CURSOR_ROLLBACK_BEHAVIOR. The value of DisableCursorHold is ignored if connecting to DB2 Server for VSE & VM where cursor with hold is not supported.

You can use this option to tune performance. It can be set to SQL_CURSOR_HOLD_OFF (1) if you are sure that your application:
1. Does not have behavior that is dependent on the SQL_CURSOR_COMMIT_BEHAVIOR or the SQL_CURSOR_ROLLBACK_BEHAVIOR information returned via SQLGetInfo(), and
2. Does not require cursors to be preserved from one transaction to the next.

The DBMS will operate more efficiently with DisableCursorHold enabled, as resources no longer need to be maintained after the end of a transaction.

## DisableDescribeParam IBM Data Server Driver configuration keyword

Disables the SQLDescribeParam() function.

**Equivalent CLI keyword**
DescribeParam

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="DisableDescribeParam" value="0 | 1"/>
```

**Default setting:**
The SQLDescribeParam() function is enabled.

**Usage notes:**

When set to 0 (default), SQLDescribeParam() is enabled and SQLGetFunctions() will return SQLDescribeParam() as supported.

When set to 1, SQLDescribeParam() is disabled. If SQLDescribeParam() is called, CLI0150E will be returned. SQLGetFunctions() will return SQLDescribeParam() as not supported.

## DisableDTCEnlist IBM Data Server Driver configuration keyword

Indicates whether automatic enlistment to the Distributed Transaction Coordinator (DTC) is enabled.

**Equivalent CLI keyword**
N/A

**Equivalent IBM Data Server Provider for .NET connection string keyword**
Enlist

**db2dsdriver.cfg configuration syntax**
```
<parameter name="DisableDTCEnlist" value="true | false"/>
```

**Default setting:**
Enlistment is allowed.

**Usage notes:**

Specifies if enlistment to Distributed Transaction Coordinator (DTC) is allowed or not:
- true = enlistment to Distributed Transaction Coordinator (DTC) is not allowed
- false = (default) enlistment to Distributed Transaction Coordinator (DTC) is allowed (this will only enlist if a COM+ transaction is in progress at connect time)

## DisablePooling IBM Data Server Driver configuration keyword

Sets the value representing whether connection pooling is disabled.

**Equivalent CLI keyword**
N/A

**Equivalent IBM Data Server Provider for .NET connection string keyword**
Pooling

**db2dsdriver.cfg configuration syntax**
```
<parameter name="DisablePooling" value="true | false"/>
```

**Default setting:**
Pooling is enabled.

**Usage notes:**

Specifies if pooling is disabled:

- true - connection pooling is disabled.
- false - connection pooling is enabled.

## DisableSynonymSchemaReporting IBM Data Server Driver configuration keyword

Controls whether CLI schema APIs report the schema name for DB2 for z/OS synonyms in the TABLE_SCHEM column of the schema procedure result sets.

**Equivalent CLI keyword**
ReturnSynonymSchema

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="DisableSynonymSchemaReporting" value="0 | 1"/>
```

**Default setting:**
**0**: By default, the creator of the synonym will be returned in the TABLE_SCHEM column.

**Usage notes:**
Valid settings:

- 0 : the TABLE_SCHEM column of the procedure result set will contain the creator of the synonym.
- 1 : the TABLE_SCHEM column of the procedure result set will be NULL.

You cannot access a synonym on a DB2 for z/OS server using a name qualified with a schema. For this reason, the meaning of the TABLE_SCHEM column of a CLI schema API result set is different, with respect to synonyms, when you are running against a DB2 for z/OS server.

This CLI keyword has no effect when you use CLI schema APIs against a DB2 Database for Linux, UNIX, and Windows server.

The following CLI APIs are affected by this keyword :

- `SQLColumns()`
- `SQLColumnPrivileges()`
- `SQLTables()`
- `SQLTablePrivileges()`
- `SQLStatistics()`
- `SQLSpecialColumns()`
- `SQLForeignKeys()`
- `SQLPrimaryKeys()`

You must have the following program temporary fixes (PTFs) on the DB2 for z/OS database server to use this keyword:

*Table 2-4. DB2 for z/OS PTFs for DisableSynonymSchemaReporting*

| DB2 for z/OS | PTF or APAR numbers |
|---|---|
| Version 7 | UK13643 |
| Version 8 | UK13644 |
| Version 9 | |

## DisableUnderscoreAsWildcard IBM Data Server Driver configuration keyword

Specifies whether the underscore character '_' is treated as a wildcard.

**Equivalent CLI keyword**
Underscore

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="DisableUnderscoreAsWildcard" value="0 | 1"/>
```

**Default setting:**
The underscore character matches any single character or no character.

**Usage notes:**

This keyword specifies if the underscore character '_' will be recognized as a wildcard or only as the underscore character. The possible settings are as follows:

- 0 - The underscore character is treated as a wildcard that matches any single character, including no character.
- 1 - The underscore character is treated only as the underscore character.

Setting DisableUnderscoreAsWildcard to 1 can improve performance when there are database objects with names that contain underscores.

This keyword applies only to the following catalog functions that accept search patterns as arguments:

- `SQLColumnPrivileges()`
- `SQLColumns()`
- `SQLProcedureColumns()`
- `SQLProcedures()`
- `SQLTablePrivileges()`
- `SQLTables()`

Note that catalog functions can only accept search patterns on particular arguments. Refer to the documentation of the specific function for details.

## DisableUnicode IBM Data Server Driver configuration keyword

Disables underlying Unicode support.

**Equivalent CLI keyword**
DisableUnicode

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="DisableUnicode" value="<not set> | 0 | 1"/>
```

**Default setting:**
Unicode support is enabled.

**Usage notes:**

With Unicode support enabled, and when called by a Unicode application, db2dsdriver will attempt to connect to the database using the best client code page possible to ensure there is no unnecessary data loss due to code

page conversion. This can increase the connection time as code pages are exchanged, or can cause code page conversions on the client that did not occur before this support was added.

If an application is Unicode (the SQL_ATTR_ANSI_APP connection attribute is set to SQL_AA_FALSE, or the connection occurred with SQLConnectW()), then the **DisableUnicode** keyword can be used to effect three different connection behaviors:

- **DisableUnicode** is not set: If the target database supports Unicode, db2dsdriver will connect in Unicode code pages (1208 and 1200). Otherwise, db2dsdriver will connect in the application code page.
- **DisableUnicode**=0 is set: db2dsdriver always connects in Unicode, whether or not the target database supports Unicode.
- **DisableUnicode**=1 is set: db2dsdriver always connects in the application code page, whether or not the target database supports Unicode.

## EnableCharToWCharMapping IBM Data Server Driver configuration keyword

Specifies the default SQL type associated with SQL_CHAR, SQL_VARCHAR, SQL_LONGVARCHAR.

**Equivalent CLI keyword**
MapCharToWChar

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="EnableCharToWCharMapping" value="0 | 1"/>
```

**Default setting:**
The default SQL type representation for SQL_CHAR, SQL_VARCHAR and SQL_LONGVARCHAR is used.

**Equivalent connection attribute:**
SQL_ATTR_MAPCHAR

**Usage notes:**
EnableCharToWCharMapping controls the SQL type that is returned when describing SQL_CHAR, SQL_VARCHAR and SQL_LONGVARCHAR columns or parameter markers.

Set EnableCharToWCharMapping as follows:

- 0 - to return the default SQL type representation
- 1 - to return SQL_CHAR as SQL_WCHAR, SQL_VARCHAR as SQL_WVARCHAR, and SQL_LONGVARCHAR as SQL_WLONGVARCHAR

Only the following DB2 CLI functions are affected by setting EnableCharToWCharMapping:

- SQLColumns()
- SQLColAttribute()
- SQLDescribeCol()
- SQLDescribeParam()
- SQLGetDescField()
- SQLGetDescRec()
- SQLProcedureColumns()

## EnableConnectionReset IBM Data Server Driver configuration keyword

Determines if the connection is placed in the connection pool upon being closed.

**Equivalent CLI keyword**
N/A

**Equivalent IBM Data Server Provider for .NET connection string keyword**
Connection Reset

**db2dsdriver.cfg configuration syntax**
```
<parameter name="EnableConnectionReset" value="true | false"/>
```

**Default setting:**
Connection is put in the connection pool when closed.

**Usage notes:**

Specifies if the connection is in the connection pool when losed:
- true - this particular connection will not be put into the connection pool when it is closed
- false - this particular connection will be put into the connection pool when it is closed

## EnableDescribeCharAsWCharForOleDb IBM Data Server Driver configuration keyword

Controls how the IBM DB2 OLE DB Provider describes CHAR, VARCHAR, LONG VARCHAR, and CLOB data.

**Equivalent CLI keyword**
OleDbReturnCharAsWChar

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="EnableDescribeCharAsWCharForOleDb" value="0 | 1"/>
```

**Default setting:**
The IBM DB2 OLE DB Provider describes CHAR, VARCHAR, LONG VARCHAR, and CLOB data as DBTYPE_WSTR.

**Usage notes:**

The IBM DB2 OLE DB Provider describes CHAR, VARCHAR, LONG VARCHAR, and CLOB data as DBTYPE_WSTR by default as of DB2 LUW Version 8.1.2. The CLI/ODBC configuration keyword EnableDescribeCharAsWCharForOleDb allows you to change this default to have the previously stated character data types reported as DBTYPE_STR.

The available settings are:
- 0 - CHAR, VARCHAR, LONG VARCHAR, and CLOB data are described as DBTYPE_STR, and the code page of data in ISequentialStream is the local code page of the client
- 1 - CHAR, VARCHAR, LONG VARCHAR, and CLOB data are reported as DBTYPE_WSTR, and the code page of data in ISequentialStream is UCS-2

## EnableKeepDynamic IBM Data Server Driver configuration keyword

Specifies if EnableKeepDynamic function is available to DB2 applications.

**Equivalent CLI keyword**
KeepDynamic

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="EnableKeepDynamic" value="0 | 1"/>
```

**Default setting:**
EnableKeepDynamic function is not available to DB2 applications.

**Equivalent connection attribute:**
SQL_ATTR_KEEP_DYNAMIC

**Usage notes:**

The EnableKeepDynamic configuration keyword should be set according to how the packages were bound on the DB2 for z/OS and OS/390 server. Set EnableKeepDynamic as follows:

- 0 - if the packages on the server were bound with the KEEPDYNAMIC NO option
- 1 - if the packages on the server were bound with the KEEPDYNAMIC YES option

It is recommended that when EnableKeepDynamic is used, the CurrentPackageSet keyword also be set. Refer to the documentation about enabling KEEPDYNAMIC support for details on how these keywords can be used together.

## EnableLobBlockingOnFetch IBM Data Server Driver configuration keyword

Enables LOB blocking fetch against servers that support LOB blocking.

**Equivalent CLI keyword**
BlockLobs

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="EnableLobBlockingOnFetch" value="0 | 1"/>
```

**Default setting:**
SQL_ATTR_BLOCK_LOBS

**Equivalent statement attribute:**
SQL_ATTR_BLOCK_LOBS

**Usage notes:**

Setting EnableLobBlockingOnFetch to 1 enables all of the LOB data associated with rows that fit completely within a single query block to be returned in a single fetch request, if the server supports LOB blocking. Clients which enable EnableLobBlockingOnFetch = 1 and bind the LOB values directly to buffers can show an increase in memory consumption depending on the amount of data retrieved for one request compared to

previous releases. LOB data is described here as being associated with a row, because the LOB data of a result set is itself not contained in the row. Instead, the row contains a reference to the actual LOB data. Therefore, with blocking of result sets returning LOB data types, any rows of the result set that fit completely within the query block (where each row consists of non-LOB data, since LOB data is not stored directly in the row), will have their associated LOB data returned from the server, if the server supports blocking of result sets returning LOB data types.

If the server does not support cursor blocking with LOB columns, then only one row of LOB data will be returned in a single fetch request and the EnableLobBlockingOnFetch value is ignored. While DB2 products support cursor blocking with LOB columns, other servers might not.

DB2 LUW does not support LOB blocking fetch.

## EnablePublicPrivelegeReporting IBM Data Server Driver configuration keyword

Reports PUBLIC privileges in `SQLColumnPrivileges()` and `SQLTablePrivileges()` results.

**Equivalent CLI keyword**
ReportPublicPrivileges

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="EnablePublicPrivilegeReporting" value="0 | 1"/>
```

**Default setting:**
PUBLIC privileges are not reported.

**Usage notes:**

This keyword specifies if privileges assigned to the PUBLIC group are to be reported as if PUBLIC was a user in the `SQLColumnPrivileges()` and `SQLTablePrivileges()` results.

## EnableSecurityInfoPersistence IBM Data Server Driver configuration keyword

Indicates if security-sensitive information, such as password, can be returned as part of the connection string after the connection has been opened or if the connection has ever been in an opened state.

**Equivalent CLI keyword**
N/A

**Equivalent IBM Data Server Provider for .NET connection string keyword**
PersistSecurityInfo

**db2dsdriver.cfg configuration syntax**

```
<parameter name="EnableSecurityInfoPersistence" value="true |
false"/>
```

**Default setting:**
Security-sensitive information is not returned

**Usage notes:**

Specifies if security-sensitive information is not returned:

- true - allow security-sensitive information, such as password, to be returned as part of the connection string after the connection has been opened or if the connection has ever been in an opened state.
- false - (default) security-sensitive information is not returned as part of the connection string. False is strongly recommended.

## EnableStaticSQLCapture IBM Data Server Driver configuration keyword

Specifies whether the application will capture SQL or use a static SQL Package for this DSN.

**Equivalent CLI keyword**
StaticMode

**Equivalent IBM Data Server Provider for .NET connection string keyword**
Not available.

**db2dsdriver.cfg configuration syntax**
```
<parameter name="EnableStaticSQLCapture" value="DISABLED | CAPTURE |
MATCH"/>
```

**Default setting:**
`Disabled` - SQL statements are not captured and no static SQL package is used.

**Usage notes:**

This option allows you to specify how the SQL issued by the application for this DSN will be processed:

- `DISABLED` = Static mode disabled. No special processing. The statements will be executed as dynamic SQL with no change. This is the default.
- `CAPTURE` = Capture Mode. Execute the statements as dynamic SQL. If the SQL statements are successful, they will be captured into a file (known as the Capture File) to be bound by the **db2cap** command later.
- `MATCH` = Match mode. Execute the CLI/ODBC statements as static SQL statements if a matching statement is found in the Capture Files specified in **`StaticPackage`**. The Capture File must first be bound by the **db2cap** command.

## EnableZOSServerMsgSP IBM Data Server Driver configuration keyword

Specifies whether a stored procedure is used to retrieve message text from DB2 for z/OS.

**Equivalent CLI keyword**
UseServerMsgSP

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="EnableZOSServerMsgSP" value="0 | 1 "/>
```

**Default setting:**
The DB2 application does not use the server stored procedure to return messages, but uses the local message files.

**Equivalent connection attribute:**
SQL_ATTR_SERVER_MSGTXT_SP

**Usage notes:**

DB2 CLI calls the stored procedure indicated by the SQL_ATTR_SERVER_MSGTXT_SP connection attribute. If this attribute is not set, DB2 CLIcalls the SYSIBM.SQLCAMESSAGE stored procedure. If this attribute is set to DSNACCMG, DB2 CLI calls DSNACCMG when connected to DB2 for z/OS Version 7 servers and calls SYSIBM.SQLCAMESSAGE when connected to DB2 for z/OS Version 8 or later.

DSNACCMG has been deprecated in DB2 for z/OS Version 9 and might be removed in a future release. If SQL_ATTR_SERVER_MSGTXT_SP is set to DSNACCMG, set this attribute to a different store procedure to retrieve messages text. Alternatively, use local message files or use the ServerMsgTextSP configuration keyword.

## ForceDescribeBeforeCall IBM Data Server Driver configuration keyword

Determines when stored procedure arguments are described.

**Equivalent CLI keyword**

DescribeCall

**Equivalent IBM Data Server Provider for .NET connection string keyword**

N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="ForceDescribeBeforeCall" value="1 | -1"/>
```

**Default setting:**

The DB2 application does not request stored procedure argument describe information when it prepares a CALL statement.

**Equivalent connection attribute:**

SQL_ATTR_DESCRIBE_CALL

**Usage notes:**

By default, the DB2 application does not request input parameter describe information when it prepares a CALL statement. If an application has correctly bound parameters to a statement, then this describe information is unnecessary and not requesting it improves performance.

The option values are:

- 1 = SQL_DESCRIBE_CALL_BEFORE. DB2 applications always requests describe information from the server, ignoring the binding information provided by the application. Setting ForceDescribeBeforeCall to 1 will also set AllowDeferredPrepare to 0 which means that describe information will also be requested for dynamic SQL statements. Note that setting AllowDeferredPrepare to 0 will not set ForceDescribeBeforeCall to 1.
- -1 = SQL_DESCRIBE_CALL_DEFAULT (default). DB2 application does not request describe information from the server and uses the binding information provided by the application. If the CALL statement execution fails, then the error recovery logic requests input parameter describe information from the server and issues the CALL statement again.

## GranteeFilter IBM Data Server Driver configuration keyword

Reduces the amount of information returned when the application gets a list of table or column privileges.

**Equivalent CLI keyword**
GranteeList

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="GranteeFilter"
value=" 'userID1', 'userID2',... 'userIDn' "/>
```

**Default setting:**
Do not filter the results.

**Usage notes:**

This option can be used to reduce the amount of information returned when the application gets a list of privileges for tables in a database, or columns in a table. The list of authorization IDs specified is used as a filter; the only tables or columns that are returned are those with privileges that have been granted *TO* those IDs.

Set this option to a list of one or more authorization IDs that have been granted privileges, delimited with single quotes, and separated by commas. For example:

```
<parameter name="GranteeFilter" value="'USER1', 'USER2', 'USER8'"/>
```

In the above example, if the application gets a list of privileges for a specific table, only those columns that have a privilege granted *TO* USER1, USER2, or USER8 would be returned.

## GrantorFilter IBM Data Server Driver configuration keyword

Reduces the amount of information returned when the application gets a list of table or column privileges.

**Equivalent CLI keyword**
GrantorList

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="GrantorFilter" value="'userID1','userID2', ...
'userIDn"/>
```

**Default setting:**
Do not filter the results.

**Usage notes:**

This option can be used to reduce the amount of information returned when the application gets a list of privileges for tables in a database, or columns in a table. The list of authorization IDs specified is used as a filter; the only tables or columns that are returned are those with privileges that have been granted *BY* those IDs.

Set this option to a list of one or more authorization IDs that have granted privileges, delimited with single quotes, and separated by commas. For example:

```
<parameter name="GrantorFilter" value="'USER1','USER2','USER8'"/>
```

In the above example, if the application gets a list of privileges for a specific table, only those columns that have a privilege granted *BY* USER1, USER2, or USER8 would be returned.

## GraphicDefaultDescribeMapping IBM Data Server Driver configuration keyword

Controls the SQL data type returned when GRAPHIC, VARGRAPHIC, and LONGVARGRAPHIC columns and parameter markers are described.

**Equivalent CLI keyword**
MapGraphicDescribe

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="GraphicDefaultDescribeMapping" value="0 | 1 | 2"/>
```

**Default setting:**
The default SQL data types are returned: SQL_GRAPHIC for GRAPHIC columns, SQL_VARGRAPHIC for VARGRAPHIC columns, and SQL_LONGVARGRAPHIC for LONG VARGRAPHIC columns.

**Usage notes:**

To control the SQL data type that is returned when GRAPHIC-based columns and parameter markers are described, set GraphicDefaultDescribeMapping as follows:

- 0 - to return the default SQL data types
- 1 - to return the CHAR-based SQL data types: SQL_CHAR for GRAPHIC columns, SQL_VARCHAR for VARGRAPHIC columns, and SQL_LONGVARCHAR for LONG VARGRAPHIC columns
- 2 - to return the WCHAR-based SQL data types: SQL_WCHAR for GRAPHIC columns, SQL_WVARCHAR for VARGRAPHIC columns, and SQL_WLONGVARCHAR for LONG VARGRAPHIC columns

Only the following DB2 CLI functions are affected by setting GraphicDefaultDescribeMapping:

- `SQLDescribeCol()`
- `SQLDescribeParam()`
- `SQLGetDescField()`
- `SQLGetDescRec()`
- `SQLProcedureColumns()`
- `SQLSpecialColumns()`

## GraphicSupportMode IBM Data Server Driver configuration keyword

Specifies if SQL_GRAPHIC (double-byte character) is returned as a supported SQL data type and what unit is used to report GRAPHIC column length.

**Equivalent CLI keyword**
Graphic

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="GraphicSupportMode" value="0 | 1 | 2 | 3"/>
```

**Default setting:**
The SQL_GRAPHIC data type is not returned as a supported SQL data type, and the length of GRAPHIC columns equals the maximum number of DBCS characters in the column.

**Usage notes:**

The GraphicSupportMode keyword controls whether the SQL_GRAPHIC (double-byte character) data type is reported as a supported SQL data type when `SQLGetTypeInfo()` is called, and what unit is used to report the length of GRAPHIC columns for all DB2 functions that return length or precision as either output arguments or as part of a result set.

Set the GraphicSupportMode keyword as follows:

- 0 - SQL_GRAPHIC is not returned as a supported SQL data type, and the reported length of GRAPHIC columns equals the maximum number of DBCS characters in the column.
- 1 - SQL_GRAPHIC is returned as a supported SQL data type, and the reported length of GRAPHIC columns equals the maximum number of DBCS characters in the column.
- 2 - SQL_GRAPHIC is not returned as a supported SQL data type, and the reported length of GRAPHIC columns equals the maximum number of bytes in the column.
- 3 - SQL_GRAPHIC is returned as a supported SQL data type, and the reported length of GRAPHIC columns equals the maximum number of bytes in the column.

## InterruptProcessingMode IBM Data Server Driver configuration keyword

Sets the interrupt processing mode.

**Equivalent CLI keyword**
Interrupt

**Equivalent IBM Data Server Provider for .NET connection string keyword**
Interrupt

**db2dsdriver.cfg configuration syntax**
```
<parameter name="InterruptProcessingMode" value="0 | 1 | 2"/>
```

**Default setting:**
1

**Usage notes:**
This can be set in the `db2dsdriver.cfg` file for the given data source.

When you set this option, you must also set the CommProtocol keyword to CommProtocol="IPC".

The keyword values have the following meaning:

**0** Disables interrupt processing (SQLCancel calls will not interrupt the processing.)

**1** Interrupts are supported (default.) In this mode, if the server supports an interrupt, an interrupt will be sent. Otherwise the connection is dropped.

The settings for INTERRUPT_ENABLED (a DB2 Connect gateway setting) and the DB2 registry variable DB2CONNECT_DISCONNECT_ON_INTERRUPT will take precedence over the Interrupt keyword setting of 1.

**2** Interrupt drops the connection regardless of server's interrupt capabilities (SQLCancel will drop the connection.)

## IPCInstance IBM Data Server Driver configuration keyword

Specifies the instance name for a local IPC connection.

**Equivalent CLI keyword**
Instance

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="IPCInstance" value="instance name"/>
```

**Default setting:**

**Usage notes:**
This can be set in the db2dsdriver.cfg file for the given data source.

When you set this keyword, you must also set the CommProtocol keyword to CommProtocol="IPC".

## IsolationLevel IBM Data Server Driver configuration keyword

Sets the default isolation level.

**Equivalent CLI keyword**
TxnIsolation

**Equivalent IBM Data Server Provider for .NET connection string keyword**
IsolationLevel

**db2dsdriver.cfg configuration syntax**
```
<parameter name="IsolationLevel" value="1 | 2 | 4 | 8 |  32"/>
```

**Default setting:**
Read Committed (Cursor Stability)

**Equivalent statement attribute:**
SQL_ATTR_TXN_ISOLATION

**Usage notes:**

Sets the isolation level to:
- 1 = SQL_TXN_READ_UNCOMMITTED - Read Uncommitted (Uncommitted read)
- 2 = SQL_TXN_READ_COMMITTED (default) - Read Committed (Cursor stability)
- 4 = SQL_TXN_REPEATABLE_READ - Repeatable Read (Read Stability)
- 8 = SQL_TXN_SERIALIZABLE - Serializable (Repeatable read)

- 32 = SQL_TXN_NOCOMMIT - (No Commit, IBM DB2 for IBM i only; this is similar to autocommit)

The words in parentheses are IBM's terminology for the equivalent SQL92 isolation levels. Note that *no commit* is not an SQL92 isolation level and is supported only on IBM DB2 for IBM i.

This keyword is only applicable if the default isolation level is used. If the application specifically sets the isolation level for a connection or statement handle, then this keyword will have no effect on that handle.

## KeepAliveTimeout IBM Data Server Driver configuration keyword

Specifies the maximum time in seconds before an unresponsive connection is detected as no longer alive.

**Equivalent CLI keyword**
N/A

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="KeepAliveTimeout" value=" 0 | 1 | 2 | ... |
32767"/>
```

**Default setting:**
The system-wide TCP/IP Keep Alive settings are used to determine if an idle TCP/IP connection is still alive.

**Usage notes:**

The value of this keyword is used to determine the number of seconds before Keep Alive probes are sent and the duration between probes. The number of seconds before probes are sent combined with the duration between probes provides an approximate maximum wait time of KeepAliveTimeout duration.

This keyword is only supported for TCP/IP protocol.

## LobAsLongDataMode IBM Data Server Driver configuration keyword

Reports LOBs as long data types or as large object types.

**Equivalent CLI keyword**
LongDataCompat

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="LobAsLongDataMode" value="0 | 1 | 2"/>
```

**Default setting:**
Reference LOB data types as large object types.

**Equivalent connection attribute:**
SQL_ATTR_LONGDATA_COMPAT

**Usage notes:**

This option indicates what data type the application expects when working with a database with large object (LOB) columns.

The values for this option are:

- 0 = Reference LOB data types as large object types.
- 1 = Report LOBs as long data types for DB2 applications only.
- 2 = Report LOBs as long data types for JDBC applications only. This does not affect applications using the DB2 Universal JDBC Driver.

*Table 2-5. Corresponding large object and long data types for LOB data*

| Database data type | Large objects (0 - Default) | Long data types (1 — CLI/ODBC; 2 — JDBC) |
|---|---|---|
| CLOB | SQL_CLOB | SQL_LONGVARCHAR |
| BLOB | SQL_BLOB | SQL_LONGVARBINARY |
| DBCLOB | SQL_DBCLOB | SQL_LONGVARGRAPHIC* |

* If the GraphicDefaultDescribeMapping keyword is set with LobAsLongDataMode, DBCLOB columns will return an SQL type of SQL_LONGVARCHAR if GraphicDefaultDescribeMapping is 1 and SQL_WLONGVARCHAR if GraphicDefaultDescribeMapping is 2.

This option is useful when running ODBC applications that cannot handle the large object data types.

The option LOBMaxColumnSize can be used with this option to reduce the default size declared for the data.

## LobCacheSize IBM Data Server Driver configuration keyword

Specifies maximum cache size (in bytes) for LOBs.

**Equivalent CLI keyword**
LOBCacheSize

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="LobCacheSize" value="positive integer"/>
```

**Default setting:**
LOBs are not cached.

**Equivalent connection or statement attribute:**
SQL_ATTR_LOB_CACHE_SIZE

**Usage notes:**

The use of LOB locators when retrieving unbound LOB data can be avoided by setting this keyword. For example, if an application does not bind a column before calling `SQLFetch()` and then calls `SQLGetData()` to fetch the LOB, if LobCacheSize was set to a value large enough to contain the entire LOB being fetched, then the LOB is retrieved from the LOB cache rather than from a LOB locator. Using the LOB cache instead of the LOB locator in this case improves performance.

Servers that support Dynamic Data Format, also known as progressive streaming, optimize the return of LOB and XML data depending on the actual length of the data. The LOB and XML data can be returned in its entirety, or as an internal token called a progressive reference. DB2 applications manage progressive reference data retrieval. The LobCacheSize defaults to 1MB if a progressive reference is possible for LOB data.

Dynamic Data Format progressive references are always used even if LOBCacheSize is not set explicitly. It will be used for any actual LOB instance that exceeds 1MB by default.

For applications that are querying data on a server that supports Dynamic Data Format, setting the LobCacheSize keyword sets a threshold that is used to determine if the data is returned in its entirety, or as a progressive reference. If the data has a length greater than the LobCacheSize threshold value, the progressive reference will be returned to the DB2 application to manage, but if the data has a length less than or equal to the LobCacheSize threshold value, the data will be returned in its entirety.

For applications that are querying data on a server that does not supports Dynamic Data Format, the LobCacheSize threshold value specifies the maximum defined size of a LOB that a DB2 application will buffer in memory. If the defined size of a LOB exceeds the value LobCacheSize is set to, then the LOB will not be cached. For example, consider a table that is created with a CLOB column of 100MB currently holding 20MB of data, with LobCacheSize set to 50MB. In this case, even though the size of the LOB itself (20MB) is less than the value set through LobCacheSize, the CLOB column will not be cached because the defined CLOB size (100MB) exceeds the maximum cache size set through LobCacheSize (50MB).

ClientBuffersUnboundLOBS is a related keyword.

## LobDataClientBufferLimit IBM Data Server Driver configuration keyword

Causes `SQLGetData()` to fetch column data in pieces of specified size (in bytes) instead of fetching column data all at once.

**Equivalent CLI keyword**
GetDataLobNoTotal

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**`db2dsdriver.cfg` configuration syntax**
`<parameter name="LobDataClientBufferLimit" value="`*positive integer*`"/>`

**Default setting:**

**Usage notes:**

`SQLGetData()` retrieves data for a single column in the current row of the result set. The first call to `SQLGetData()` does the following:

- Fetches all the data from the database server to the client, which requires allocating memory on the client for the data
- Applies a code page conversion to that data, if required
- Calculates the total length of the converted data
- Returns the length of the converted data to the client application

When the data is very large, allocating memory for the data on the client might fail. You can avoid this potential memory allocation problem, by using the LobDataClientBufferLimit keyword.

When you set the LobDataClientBufferLimit keyword, `SQLGetData()` does not fetch all the data for the given column on the first call. Instead, `SQLGetData()` fetches enough data to fill the buffer on the client, as specified by the value of LobDataClientBufferLimit, and returns

SQL_NO_TOTAL (-4) if there is more data to be fetched from the server. You can call SQLGetData() as many times as needed to fetch all the data. When all the data has been fetched, SQLGetData() returns SQL_SUCCESS and the size of the last data chunk.

## LobMaxColumnSize IBM Data Server Driver configuration keyword

Overrides the default value in the COLUMN_SIZE column for LOB data types.

**Equivalent CLI keyword**
LOBMaxColumnSize

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="LobMaxColumnSize" value="integer greater than zero"/>
```

**Default setting:**
2 Gigabytes (1G for DBCLOB)

**Usage notes:**

This will override the 2 Gigabyte (1G for DBCLOB) value that is returned by `SQLGetTypeInfo()` for the COLUMN_SIZE column for SQL_CLOB, SQL_BLOB, and SQL_DBCLOB and SQL_XML SQL data types. For SQL_XML, LobMaxColumnSize must be specified with XMLDefaultDescribeMapping set to a LOB type. Subsequent CREATE TABLE statements that contain LOB columns will use the column size value you set here instead of the default.

## LockTimeout IBM Data Server Driver configuration keyword

Sets the default value of the LOCKTIMEOUT configuration parameter.

**Equivalent CLI keyword**
LockTimeout

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="LockTimeout" value="-1 | 0 | positive integer ≤ 32767"/>
```

**Default setting:**
Timeout is turned off (-1), with the application waiting for a lock until either the lock is granted or deadlock occurs.

**Usage notes:**

The LockTimeout keyword specifies the number of seconds a DB2 application will wait to obtain locks. If the keyword is set to 0, locks will not be waited for. The -1 setting causes the application to wait indefinitely until either the lock is granted or deadlock occurs.

## MaxPoolSize IBM Data Server Driver configuration keyword

Specifies the maximum connection pool size.

**Equivalent CLI keyword**
N/A

**Equivalent IBM Data Server Provider for .NET connection string keyword**
Max Pool Size

**db2dsdriver.cfg configuration syntax**
```
<parameter name="MaxPoolSize" value="no maximum"/>
```

**Default setting:**
There is no maximum.

**Usage notes:**
An integer representing the maximum number of connections in the connection pool. The minimum pool size is set with MinPoolSize.

## MinPoolSize IBM Data Server Driver configuration keyword

Specifies the minimum connection pool size.

**Equivalent CLI keyword**
N/A

**Equivalent IBM Data Server Provider for .NET connection string keyword**
Min Pool Size

**db2dsdriver.cfg configuration syntax**
```
<parameter name="MinPoolSize" value="0"/>
```

**Default setting:**
There default size is zero.

**Usage notes:**
An integer representing the minimum number of connections in the connection pool. The maximum pool size is set with MaxPoolSize.

## NumRowsOnFetch IBM Data Server Driver configuration keyword

Specifies the number of rows of data to be returned in a single fetch.

**Equivalent CLI keyword**
BlockForNRows

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="NumRowsOnFetch" value="<positive integer>"/>
```

**Default setting:**
The server returns as many rows as can fit in a query block in a single fetch request.

**Usage notes:**

The NumRowsOnFetch keyword controls the number of rows of data that are returned to the client in a single fetch request. If NumRowsOnFetch is not specified (the default setting), then as many rows of non-LOB data as can fit in a query block are returned from the server. If the result set contains LOB data, then the behavior NumRowsOnFetch yields can be affected by the EnableLobBlockingOnFetch dsdriver configuration keyword and the server's support for blocking of result sets returning LOB data types.

All LOB data associated with rows that fit completely within a single query block are returned in a single fetch request if:

- NumRowsOnFetch is not specified,
- EnableLobBlockingOnFetch is set to 1 and
- the server supports blocking of result sets returning LOB data types.

LOB data is described here as being associated with a row, because the LOB data of a result set is itself not contained in the row. Instead, the row contains a reference to the actual LOB data.

If NumRowsOnFetch is set to a positive integer n, then n rows of data will be returned in a single fetch request. If the result set contains LOB data and the server supports blocking of result sets returning LOB data types, then the LOB data that corresponds to the n rows of data will also be returned in the single fetch request. If the result set contains LOB data, but the server does not support blocking of result sets returning LOB data types, then only one row of data, including the LOB data, will be returned in a single fetch request.

## ProgramName IBM Data Server Driver configuration keyword

Sets the default client application name to a user-defined name which is used to identify the application at the server when monitoring.

**Equivalent CLI keyword**
ProgramName

**Equivalent IBM Data Server Provider for .NET connection string keyword**
ProgramName

**db2dsdriver.cfg configuration syntax**

```
<parameter name="ProgramName" value="<string> | PID"/>
```

**Default setting:**
The application is identified by the client. By default, the first 20 bytes of the executable name is used.

**Equivalent connection attribute:**
SQL_ATTR_INFO_PROGRAMNAME

**Usage notes:**

When monitoring an application, it can be useful to identify the application by a user-defined string, instead of by the default identifier that DB2 assigns. ProgramName allows the user to specify the identifier as either a string up to 20 bytes in length or the string "PID" (without the quotation marks).

If ProgramName is set to "PID" for an application, the application's name will consist of the a prefix along with the application's process ID and the current active connection handle, as follows: <prefix><pid>:<connectionHandle#>. The "PID" setting is useful when monitoring application servers that run multiple applications with numerous connections to the same database.

(When the ProgramName keyword is set to "PID" for other types of applications, the application prefix is replaced with the following values corresponding to the type of application: "JDBC" for JDBC applications, "OLEDB" for OLE DB applications, and "ADONET" for .NET applications.)

## QueryOptimizationLevel IBM Data Server Driver configuration keyword

Sets the query optimization level.

**Equivalent CLI keyword**
DB2Optimization

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="QueryOptimizationLevel" value="integer value from 0 to 9"/>
```

**Default setting:**
No SET CURRENT QUERY OPTIMIZATION statement issued.

**Usage notes:**

If this option is set, a DB2 application issues the following SQL statement after a successful connection:

```
SET CURRENT QUERY OPTIMIZATION positive number
```

This specifies the query optimization level at which the optimizer should operate the SQL queries.

## QueryTimeoutInterval IBM Data Server Driver configuration keyword

Delay (in seconds) between checks for a query timeout.

**Equivalent CLI keyword**
QueryTimeoutInterval

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="QueryTimeoutInterval" value="0 | 5 | positive integer"/>
```

**Default setting:**
5 seconds

**Usage notes:**

An application can use the `SQLSetStmtAttr()` function to set the SQL_ATTR_QUERY_TIMEOUT statement attribute. This attribute indicates the number of seconds to wait for an SQL statement or XQuery expression to complete executing before attempting to cancel the execution and returning to the application.

The QueryTimeoutInterval configuration keyword is used to indicate how long the CLI driver should wait between checks to see if the query has completed.

For instance, suppose SQL_ATTR_QUERY_TIMEOUT is set to 25 seconds (timeout after waiting for 25 seconds), and QueryTimeoutInterval is set to 10 seconds (check the query every 10 seconds). The query will not time out until 30 seconds (the first check AFTER the 25 second limit).

**Note:** DB2 applications implement query timeout by starting a thread that periodically queries the status of each executing query. The QueryTimeoutInterval value specifies how long the query timeout thread waits between checks for expired queries. Because this is an asynchronous operation to the queries being run, it is possible that a given query may

not be timed out until SQL_ATTR_QUERY_TIMEOUT + QueryTimeoutInterval seconds. In the example above, the best-case timeout would be at 26 seconds, and the worst-case timeout would be at 35 seconds.

There might be cases where the SQL_ATTR_QUERY_TIMEOUT is set to a value which is too low, and the query should NOT be timed-out. If the application cannot be modified (that is, a third party ODBC application), then the QueryTimeoutInterval can be set to 0, and the dsdriver driver will ignore the SQL_ATTR_QUERY_TIMEOUT setting, and therefore wait for SQL statements to complete execution before returning to the application.

**Note:** If QueryTimeoutInterval is set to 0, any attempt by the application to set SQL_ATTR_QUERY_TIMEOUT will result in SQLSTATE 01S02 (Option Value Changed).

(This option is contained in the Common section of the initialization file and therefore applies to all DB2 connections.

Alternatively, QueryTimeoutInterval can be set to a value that is larger than the SQL_ATTR_QUERY_TIMEOUT setting, thus preventing time-outs from occurring at the specified interval. For example, if the application sets a 15 second SQL_ATTR_QUERY_TIMEOUT value, but the server requires at least 30 seconds to run the query, the QueryTimeoutInterval can be set to a value of 30 seconds or so to prevent this query from timing out after 15 seconds.

## ReceiveTimeout IBM Data Server Driver configuration keyword

Specifies the time in seconds to wait for a reply from the server on an established connection before terminating the attempt and generating a communication timeout error.

**Equivalent CLI keyword**
ReceiveTimeout

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="ReceiveTimeout" value="0 | 1 |2 | ... | 32767"/>
```

**Default setting:**
The client waits indefinitely for a reply from the server on an established connection.

**Equivalent connection attribute:**
SQL_ATTR_RECEIVE_TIMEOUT

**Usage notes:**

This keyword has no effect during connection establishment and is only supported for TCP/IP protocol.

## SchemaFilter IBM Data Server Driver configuration keyword

Restricts schemas used to query table information.

**Equivalent CLI keyword**
SchemaList

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="SchemaFilter"
value=" 'schema1', 'schema2',... 'schemaN' "/>
```

**Default setting:**

None

**Usage notes:**

SchemaFilter is used to provide a more restrictive default, and therefore improve performance, for those applications that list every table in the DBMS.

If there are a large number of tables defined in the database, a schema list can be specified to reduce the time it takes for the application to query table information, and reduce the number of tables listed by the application. Each schema name is case-sensitive, must be delimited with single quotation marks, and separated by commas. For example:

```
SchemaFilter='USER1','USER2','USER3'
```

For DB2 for z/OS, CURRENT SQLID can also be included in this list, but without the single quotation marks, for example:

```
SchemaFilter='USER1',CURRENT SQLID,'USER3'
```

The maximum length of the string is 256 characters.

This option can be used in conjunction with ZOSDBNameFilter and TableTypeFilter to further limit the number of tables for which information will be returned.

## SecurityTransportMode IBM Data Server Driver configuration keyword

Sets the value representing the security type being used.

**Equivalent CLI keyword**

N/A

**Equivalent IBM Data Server Provider for .NET connection string keyword**

Security

**db2dsdriver.cfg configuration syntax**

```
<parameter name="SecurityTransportMode" value="SSL"/>
```

**Default setting:**

There is no default setting.

**Usage notes:**

Set to SSL if use of Secure Sockets Layer (SSL) is enabled and is to be used as a secure transport. The default value is an empty string.

## ServerType IBM Data Server Driver configuration keyword

This property applies to UniData® and UniVerse data servers only. Indicates whether the connection is to be made to UniData or UniVerse.

**Equivalent CLI keyword**

N/A

**Equivalent IBM Data Server Provider for .NET connection string keyword**

ServerType

**db2dsdriver.cfg configuration syntax**

```
<parameter name="ServerType" value="UNIDATA"/>
```

**Default setting:**

The default value is an empty string ("")

**Usage notes:**

Specifies the data server type to connect to. Use only for UniData and UniVerse data servers, you must specify a ServerType property with a value of UNIDATA or UNIVERSE in the ConnectionString.

## SkipSynonymProcessing IBM Data Server Driver configuration keyword

Specifies whether IBM .NET Data Provider for .NET sends a connection string as is to the DbPermission.Add method.

**Equivalent CLI keyword**

N/A

**Equivalent IBM Data Server Provider for .NET connection string keyword**

SkipSynonymProcessing

**db2dsdriver.cfg configuration syntax**

```
<parameter name="SkipSynonymProcessing" value="true | false | yes |
no | 0 | 1"/>
```

**Default setting:**

false

**Usage notes:**

The value of the keyword determines whether synonym processing succeeds.

The possible values for the keyword are as follows:

**true | yes | 1**

Indicates that the connection string is not processed for synonyms.

**false | no | 0**

Indicates that the connection string is processed for synonyms.

## StatementConcentrator IBM Data Server Driver configuration keyword

Sets whether or not statement concentrator literals are enabled

### Equivalent CLI keyword

StatementConcentrator

### Equivalent IBM Data Server Provider for .NET connection string keyword

**StatementConcentrator**

### db2dsdriver.cfg configuration syntax

```
<parameter name="StatementConcentrator" value="Off | Literals"/>
```

### Default setting:

The default behavior when this keyword is not set is defined by the server-side configuration.

### Equivalent statement attribute:

N/A

### Usage notes:

The **StatementConcentrator** keyword overrides the server configuration for statement concentrator literals in the following way:

- `Off` - statement concentrator literals are disabled.
- `Literals` - statement concentrator literals are enabled. Literals will be converted to parameters.

### Remarks

When you use this attribute against DB2 for z/OS servers older than v10, the request is ignored.

## StaticSQLCaptureFile IBM Data Server Driver configuration keyword

Specifies the Capture File name and optionally the path where it will be saved.

**Equivalent CLI keyword**
StaticCapFile

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="StaticSQLCaptureFile" value="< Full file name >"/>
```

**Default setting:**
None - you must specify a capture file.

**Usage notes:**

This keyword is used to specify the Capture File name and optionally the directory where it will be saved.

## TableTypeFilter IBM Data Server Driver configuration keyword

Defines a default list of TABLETYPES returned when querying table information.

**Equivalent CLI keyword**
TableType

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="TableTypeFilter"
value=" 'TABLE' | ,'ALIAS' | ,'VIEW' | , 'INOPERATIVE VIEW' | ,
'SYSTEM TABLE' | ,'SYNONYM' "/>
```

**Default setting:**
No default list of TABLETYPES is defined.

**Usage notes:**

If there is a large number of tables defined in the database, a tabletype string can be specified to reduce the time it takes for the application to query table information, and reduce the number of tables listed by the application.

Any number of the values can be specified. Each type must be delimited with single quotation marks, separated by commas, and in uppercase. For example:

```
TableTypeFilter='TABLE','VIEW'
```

This option can be used in conjunction with ZOSDBNameFilter and SchemaFilter to further limit the number of tables for which information will be returned.

TableTypeFilter is used to provide a default for the DB2 function that retrieves the list of tables, views, aliases, and synonyms in the database. If the application does not specify a table type on the function call, and this keyword is not used, information about all table types is returned. If the application does supply a value for the *tabletype* on the function call, then that argument value will override this keyword value.

If TableTypeFilter includes any value other than TABLE, then the ZOSDBNameFilter keyword setting cannot be used to restrict information to a particular DB2 for z/OS database.

## TimeDefaultCMapping IBM Data Server Driver configuration keyword

Specifies the default C type of TIME columns and parameter markers.

**Equivalent CLI keyword**
MapTimeCDefault

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="TimeDefaultCMapping" value="0 | 1 | 2"/>
```

**Default setting:**
The default C type representation for TIME data is SQL_C_TYPE_TIME.

**Usage notes:**

TimeDefaultCMapping controls the C type that is used when SQL_C_DEFAULT is specified for TIME columns and parameter markers. This keyword should be used primarily with Microsoft applications, such as Microsoft Access, which assume SQL_C_CHAR as the default C type for datetime values. Set TimeDefaultCMapping as follows:

- 0 - for the default SQL_C_TYPE_TIME C type representation: a struct containing numeric members for hour, minute, and second
- 1 - for an SQL_C_CHAR C type representation: "12:34:56"
- 2 - for an SQL_C_WCHAR C type representation: "12:34:56" in UTF-16.

This keyword affects the behavior of CLI functions where SQL_C_DEFAULT can be specified as a C type, such as `SQLBindParameter()`, `SQLBindCol()`, and `SQLGetData()`.

**Note:** TimeDefaultCMapping supersedes CLIPatch2=24. If both TimeDefaultCMapping and CLIPatch2=24 are set, the TimeDefaultCMapping value takes precedence.

## TimeDefaultDescribeMapping IBM Data Server Driver configuration keyword

Controls the SQL data type returned when TIME columns and parameter markers are described.

**Equivalent CLI keyword**
MapTimeDescribe

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="TimeDefaultDescribeMapping" value="0 | 1 | 2"/>
```

**Default setting:**
The default SQL data type for TIME data is returned: SQL_TIME for ODBC 2.0 or SQL_TYPE_TIME for ODBC 3.0.

**Usage notes:**

To control the SQL data type that is returned when TIME columns and parameter markers are described, set TimeDefaultDescribeMapping as follows:

- 0 - to return the default SQL data type: SQL_TIME for ODBC 2.0 or SQL_TYPE_TIME for ODBC 3.0
- 1 - to return the SQL_CHAR SQL data type
- 2 - to return the SQL_WCHAR SQL data type

Only the following DB2 functions are affected by setting TimeDefaultDescribeMapping:

- `SQLColumns()`
- `SQLDescribeCol()`
- `SQLDescribeParam()`
- `SQLGetDescField()`
- `SQLGetDescRec()`
- `SQLProcedureColumns()`
- `SQLSpecialColumns()`

## TimestampDefaultCMapping IBM Data Server Driver configuration keyword

Specifies the default C type of TIMESTAMP columns and parameter markers.

**Equivalent CLI keyword**
MapTimestampCDefault

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="TimestampDefaultCMapping" value="0 | 1 | 2"/>
```

**Default setting:**
The default C type representation for TIMESTAMP data is SQL_C_TYPE_TIMESTAMP.

**Usage notes:**

TimestampDefaultCMapping controls the C type that is used when SQL_C_DEFAULT is specified for TIMESTAMP columns and parameter markers. This keyword should be used primarily with Microsoft applications, such as Microsoft Access, which assume SQL_C_CHAR as the default C type for datetime values. Set TimestampDefaultCMapping as follows:

- 0 - for the default SQL_C_TYPE_TIMESTAMP C type representation: a struct containing numeric members for year, month, day, hour, minute, second, and fraction of a second
- 1 - for an SQL_C_CHAR C type representation: "2008-01-01 12:34:56.123456"
- 2 - for an SQL_C_WCHAR C type representation: "2008-01-01 12:34:56.123456" in UTF-16.

This keyword affects the behavior of CLI functions where SQL_C_DEFAULT can be specified as a C type, such as `SQLBindParameter()`, `SQLBindCol()`, and `SQLGetData()`.

## TimestampDefaultDescribeMapping IBM Data Server Driver configuration keyword

Controls the SQL data type returned when TIMESTAMP columns and parameter markers are described.

**Equivalent CLI keyword**
MapTimestampDescribe

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**`db2dsdriver.cfg` configuration syntax**
```
<parameter name="TimestampDefaultDescribeMapping" value="0 | 1 |
2"/>
```

**Default setting:**
The default SQL data type for TIMESTAMP data is returned: SQL_TIMESTAMP for ODBC 2.0 or SQL_TYPE_TIMESTAMP for ODBC 3.0.

**Usage notes:**

To control the SQL data type that is returned when TIMESTAMP columns and parameter markers are described, set TimestampDefaultDescribeMapping as follows:

- 0 - to return the default SQL data type: SQL_TIMESTAMP for ODBC 2.0 or SQL_TYPE_TIMESTAMP for ODBC 3.0
- 1 - to return the SQL_CHAR SQL data type
- 2 - to return the SQL_WCHAR SQL data type

Only the following DB2 functions are affected by setting TimestampDefaultDescribeMapping:

- `SQLColumns()`
- `SQLDescribeCol()`

- SQLDescribeParam()
- SQLGetDescField()
- SQLGetDescRec()
- SQLProcedureColumns()
- SQLSpecialColumns()

## TrustedContextSystemPassword IBM Data Server Driver configuration keyword

Specifies the password to be used with the connection.

**Equivalent CLI keyword**
Trusted_Connection

**Equivalent IBM Data Server Provider for .NET connection string keyword**
TrustedContextSystemPassword

**db2dsdriver.cfg configuration syntax**
```
<parameter name="TrustedContextSystemPassword" value="<password>"/>
```

**Default setting:**
DB2 applications use the user ID and password information provided in the connection string to SQLDriverConnect(), not the current authenticated user.

**Usage notes:**
Specifies the password corresponding to the trusted context SYSTEM AUTHID to be used with the connection.

## TrustedContextSystemUserID IBM Data Server Driver configuration keyword

Specifies the ID to be used with the connection.

**Equivalent CLI keyword**
Trusted_Connection

**Equivalent IBM Data Server Provider for .NET connection string keyword**
TrustedContextSystemUserID

**db2dsdriver.cfg configuration syntax**
```
<parameter name="TrustedContextSystemUserID" value="<userid>"/>
```

**Default setting:**
DB2 applications use the user ID and password information provided in the connection string to SQLDriverConnect(), not the current authenticated user.

**Usage notes:**
Specifies the trusted context SYSTEM AUTHID to be used with the connection.

## UserID IBM Data Server Driver configuration keyword

Defines a default user ID.

**Equivalent CLI keyword**
UID

**Equivalent IBM Data Server Provider for .NET connection string keyword**
UID

**db2dsdriver.cfg configuration syntax**

```
<parameter name="UserID" value="userid"/>
```

**Default setting:**

None

**Usage notes:**

The specified *userid* value is used if a userid is not provided by the application at connect time.

## XMLDeclarationGenMode IBM Data Server Driver configuration keyword

Controls the generation of an XML declaration when XML data is implicitly serialized to an application variable.

**Equivalent CLI keyword**

XMLDeclaration

**Equivalent IBM Data Server Provider for .NET connection string keyword**

N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="XMLDeclarationGenMode" value="non-negative integer
< 7 | 7"/>
```

**Default setting:**

A BOM and an XML declaration containing the XML version and encoding attribute are generated during implicit serialization.

**Usage notes:**

The XMLDeclaration keyword controls which elements of an XML declaration are prepended to an application buffer when XML data is implicitly serialized to an application buffer. This setting does not affect the result of the XMLSERIALIZE function.

The following values represent components to be generated during implicit serialization. Set this keyword by adding together the value of each component required.

**0** No declarations or byte order marks (BOMs) are added to the output buffer.

**1** A byte order mark (BOM) in the appropriate endianness is prepended to the output buffer if the target encoding is UTF-16 or UTF-32. (Although a UTF-8 BOM exists, the database server does not generate it, even if the target encoding is UTF-8.)

**2** A minimal XML declaration is generated, containing only the XML version.

**4** An encoding attribute that identifies the target encoding is added to any generated XML declaration. Therefore, this setting only has effect when the setting of 2 is also included when computing the value of this keyword.

For example, if you wanted a BOM and minimal XML declaration (without an encoding attribute) to be generated during implicit serialization, set `XMLDeclaration = 3`, where 3 is the sum of 1 (the value to indicate generation of a BOM) and 2 (the value to indicate generation of a minimal XML declaration).

To prevent any declarations or BOM from being generated, set XMLDeclaration as follows: `XMLDeclaration = 0`.

## XMLDefaultCMapping IBM Data Server Driver configuration keyword

Controls the default C type representation used when SQL_C_DEFAULT is specified for XML columns and parameter markers.

**Equivalent CLI keyword**
MapXMLCDefault

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="XMLDefaultCMapping" value="0 | 1 | 2 | 3"/>
```

**Default setting:**
The default C type representation for XML data is SQL_C_BINARY.

**Usage notes:**

XMLDefaultCMapping controls the C type that is used when SQL_C_DEFAULT is specified for XML columns and parameter markers. Use this keyword primarily with Microsoft applications, such as Microsoft Access, which might assume SQL_C_WCHAR as the default C type for XML values. Set XMLDefaultCMapping as follows:

- 0 - for the default SQL_C_BINARY C type representation
- 1 - for the SQL_C_CHAR C type representation; this can result in data loss as the XML data is converted to the local application code page
- 2 - for the SQL_C_WCHAR C type representation

This keyword affects the behavior of CLI functions where SQL_C_DEFAULT can be specified as a C type, such as `SQLBindParameter()`, `SQLBindCol()`, and `SQLGetData()`.

## XMLDefaultDescribeMapping IBM Data Server Driver configuration keyword

Controls the SQL data type returned when XML columns and parameter markers are described.

**Equivalent CLI keyword**
MapXMLDescribe

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**

```
<parameter name="XMLDefaultDescribeMapping" value="-370 | -350 |
-152 | -99 | -98"/>
```

**Default setting:**
The default SQL data type for XML data is returned: SQL_XML (-370)

**Usage notes:**

To control the SQL data type that is returned when XML columns and parameter markers are described, set XMLDefaultDescribeMapping to one of the following integer values:

- -370 to return the default SQL_XML SQL data type

- -350 to return the SQL_DBCLOB SQL data type
- -152 to return the SQL_SS_XML SQL data type

  **Note:** The SQL_SS_XML value of -152 belongs to the reserved range of Microsoft SQL Server and is not defined by IBM.
- -99 to return the SQL_BLOB SQL data type
- -98 to return the SQL_CLOB SQL data type

The data length for XML values mapped to LOB types is the maximum length for the mapped data type.

When used with the LobAsLongDataMode keyword set to the value 1, XML values mapped to LOB data types will be mapped to the corresponding LONG data type as well.

Character types specified for XMLDefaultDescribeMapping can result in data loss during data conversion if the application code page does not support all of the characters in the source data. Mapping XML values to character types, therefore, is only recommended with caution.

This keyword is recommended to provide compatibility with applications that access XML columns as CLOB or BLOB, or use Microsoft application development technologies.

## ZOSDBNameFilter IBM Data Server Driver configuration keyword

Specifies the database name to reduce the time it takes for the application to query z/OS or OS/390 table information.

**Equivalent CLI keyword**
DBName

**Equivalent IBM Data Server Provider for .NET connection string keyword**
N/A

**db2dsdriver.cfg configuration syntax**
```
<parameter name="ZOSDBNameFilter" value="dbname"/>
```

**Default setting:**
Do not filter on the DBNAME column.

**Usage notes:**

This option is only used when connecting to DB2 for z/OS and OS/390, and only if (*base*) table catalog information is requested by the application. If a large number of tables exist in the z/OS or OS/390 subsystem, a *dbname* can be specified to reduce the time it takes for the application to query table information, and reduce the number of tables listed by the application.

If this option is set then the statement IN DATABASE *dbname* will be appended to various statements such as CREATE TABLE.

This value maps to the `DBNAME` column in the z/OS or OS/390 system catalog tables. If no value is specified, or if views, synonyms, system tables, or aliases are also specified via TableTypeFilter, only table information will be restricted; views, aliases, and synonyms are not restricted with ZOSDBNameFilter. It can be used in conjunction with SchemaFilter, and TableTypeFilter to further limit the number of tables for which information will be returned.

## Installing IBM Data Server Driver Package (Windows)

On Windows operating systems, IBM Data Server Driver Package simplifies application deployment. This driver, which has a small footprint, is designed to be redistributed by independent software vendors (ISVs) and to be used for application distribution in mass deployment scenarios typical of large enterprises.

IBM Data Server Driver Package is not part of IBM Data Server Client or IBM Data Server Runtime Client. These are different packages and they have their own install images. You need to use the appropriate installer package.

IBM Data Server Driver Package must be installed separately. No other DB2 database product can be installed in the same path if IBM Data Server Driver Package is already installed.

IBM Data Server Driver Package is installed by running the DB2 setup program from the product DVD. There are separate install images for each language.

To install IBM Data Server Driver Package using a response file, or to install it from a fix pack image, you can run the DB2 **setup** command from the command line. A sample response file is located in the `\samples` subdirectory. For fix pack images, you can download the driver package that contains the setup program from the IBM Support Fix Central Web site: http://www.ibm.com/support/fixcentral/. Data Server client and driver packages are found under the **Information Management** product group and **IBM Data Server Client Packages** product selection. Select the appropriate installed version and platform, in this case Windows, and click **Continue**. Click **Continue** again on the next screen and you will be presented with a list of all client and driver packages available for Windows.

The default installation path of IBM Data Server Driver Package is `Program Files\IBM\IBM DATA SERVER DRIVER`. If a second copy is installed in the same machine, the default directory name is `Program Files\IBM\IBM DATA SERVER DRIVER_02`. In general, the default directory name is `IBM DATA SERVER DRIVER_`*nn* where *nn* is the generated number to make this directory unique.

If you want to install more than one copy of IBM Data Server Driver Package, you can have a maximum of 16 copies. Each copy must be installed to different directories. The default copy name of IBM Data Server Driver Package is IBMDBCL1. If there are further drivers are installed on your machine, they will receive default names: IBMDBCL2, IBMDBCL3, and so on.

**Note:** Installation of multiple copies is an advanced installation method that is not recommended for most users.

The db2diag.log path for IBM Data Server Driver Package on Windows is *%SYSTEM APP DATA PATH%*`\IBM\DB2\COPYNAME` , where *%SYSTEM APP DATA PATH%* is `Drive:\Documents and Settings\All Users\Application Data\` for Windows 2003, Windows XP and `Drive:\ProgramData\` for Windows 2008 and Windows Vista. The `COPYNAME` is the name of your IBM Data Server Driver Package copyname provided during the installation.

The DB2_DIAGPATH environment variable and the DIAGPATH CLI keyword can be used to alter the location of the db2diag log files.

After installing IBM Data Server Driver Package, you can optionally create and populate the configuration file, `db2dsdriver.cfg`, with database directory information.

## Installing IBM data server clients (Windows)

Instructions to install any IBM data server client type, namely the IBM Data Server Client, the IBM Data Server Runtime Client, and the IBM Data Server Driver Package. The main procedure covers a simple, but common, case where there is no DB2 database product already installed.

If the machine already has a prior version of a client installed, you should first review topics that cover upgrading.

If the machine already has a DB2 database server product installed, it is not necessary to install a client because the DB2 database server provides all the capability found in an IBM data server client.

Before installing IBM data server clients or client package:

- You have determined which client best suits your need.
- You have located a DVD or other install image that you need. You can download an image from the IBM Support Fix Central Web site: http://www.ibm.com/support/fixcentral/. Data Server client and driver packages are found under the **Information Management** product group and **IBM Data Server Client Packages** product selection. Select the appropriate installed version and platform and click **Continue**. Click **Continue** again on the next screen and you will be presented with a list of all client and driver packages available for Windows. Ensure you have the appropriate 32-bit or 64-bit version, depending on your machine.
- You have a Windows user account that is part of the Administrators group.

  **Note:** If a non-Administrator user account is going to do the product installation, then the VS2005 runtime library must be installed before attempting to install a DB2 database product. The VS2005 runtime library is needed on the operating system before the DB2 database product can be installed. The VS2005 runtime library is available from the Microsoft runtime library download web site. There are two choices: choose `vcredist_x86.exe` for 32-bit systems or `vcredist_x64.exe` for 64-bit systems.
- Your system meets all memory, disk space, and installation requirements. The installation program will check disk space and basis system requirements, and notify you if there is a problem.

**Restrictions**

- No other DB2 database product can be installed in the same path if one of the following products is already installed:
  - IBM Data Server Runtime Client
  - IBM Data Server Driver Package
  - *DB2 Information Center*
- The DB2 Setup wizard fields do not accept non-English characters.

This procedure covers the simple case. Information for other cases is covered elsewhere in this topic.

To install any IBM data server client on Windows:

1. Log on to the system with the user account that you want to use to perform the installation.
2. Optional: Shut down any other programs.
3. Insert the DVD into the drive. The autorun feature starts the DB2 Setup wizard which determines the system language and starts the setup program for that language.

   For the Data Server Client, you can run the DB2 Setup wizard in a language other than the default system language by manually invoking the DB2 Setup wizard and specifying a language code. For example, the **setup -i fr** command runs the DB2 Setup wizard in French. For the Data Server Runtime Client or
1 Data Server Driver Package, there are separate install images for each language.
4. To install an IBM data server client or client package:
   - If you are installing a Data Server Client, launch the DB2 Setup wizard, when the DB2 Launchpad opens, choose **Install a Product**. Follow the DB2 Setup wizard's prompts.
   - If you are installing a Data Server Runtime Client, it does not have a launchpad. See the Related Links for **setup** command parameters.

     If you are installing a second copy of the Data Server Runtime Client, the command is:

     ```
     setup /v" TRANSFORMS=:InstanceId1.mst MSINEWINSTANCE=1"
     ```

     To install each subsequent copy of the Data Server Runtime Client (up to a maximum of 16 copies), modify the command by incrementing `InstanceId`*n*, for example:

     ```
     setup /v" TRANSFORMS=:InstanceId2.mst MSINEWINSTANCE=1"
     ```

3 3 **Note:** It is strongly recommended that installing multiple copies is for advanced users.

1 - If you are installing Data Server Driver Package, run the **setup** command from the product DVD, or install the driver from a fix pack image by downloading the driver from http://www.ibm.com/support/docview.wss?rs=71&uid=swg27007053. If you are installing a fix pack image, see the Related Links for the installation command options of the **setup** command.

1 If you are installing a second copy of Data Server Driver Package, the following methods can be used:
     - To perform a new copy installation with a generated default copy name:

       ```
       setup /o
       ```
     - If the copy name already exists, perform a maintenance (or upgrade) installation on that copy. Otherwise, perform the new installation by using the specified copy name.

       ```
       setup /n copyname
       ```

1 After installing Data Server Driver Package, you can optionally create and populate the configuration file, `db2dsdriver.cfg`, with database directory information.
5. If you are installing a Data Server Client on a machine that already has a DB2 Universal Database (UDB) Version 8 copy installed, you will be presented with the option to install a new copy or to upgrade the DB2 UDB Version 8 copy.

   Installing a new copy preserves the DB2 UDB Version 8 copy and installs an additional DB2 Version 9 copy. Choosing to upgrade will copy the DB2 UDB Version 8 client instance settings to the DB2 Version 9 copy then remove the DB2 UDB Version 8 copy.

**Note:** If a machine already has a DB2 UDB Version 8 copy installed, the Version 9 copies cannot be set to default.

If you are installing a Data Server Runtime Client, the installation program always installs a new copy. To upgrade a DB2 UDB Version 8 client instance as a subsequent step, see topics on migration.

After completing this procedure, the product is now installed at the location you specified during the installation. As part of the IBM Data Server Client installation procedure, an instance of the DB2 database manager is created. The instance is called "DB2" if there is no other instance called "DB2". If you already have a copy of DB2 Version 8 or DB2 Version 9.1 installed, the default instance is DB2_01.

The default installation path of Data Server Client and Data Server Runtime Client is `Program Files\IBM\sqllib`. If a second copy is installed in the same machine, the default directory name is `Program Files\IBM\sqllib_01`. In general, the default directory name is `sqllib_`*nn* where *nn* is the number of copies installed in that machine minus one.

1 The default installation path of Data Server Driver Package is `Program Files\IBM\IBM DATA SERVER DRIVER`. If a second copy is installed in the same machine, the default directory name is `Program Files\IBM\IBM DATA SERVER DRIVER_02`. In general, the default directory name is `IBM DATA SERVER DRIVER_`*nn* where *nn* is the generated number to make this directory unique.

1 If you want to install more than one copy of the Data Server Driver Package, you can have a maximum of 16 copies. Each copy must be installed to different directories.

The default copy name of the Data Server Client or Data Server Runtime Client is `DB2COPY1`

1 The default copy name of the Data Server Driver Package is `IBMDBCL1`

This installation does not include product documentation. See the related links for options for installing or accessing the *DB2 Information Center*.

After installing your IBM data server client, the next step is to configure it to access remote DB2 database servers.

**Notes on installing using a user account that is not a member of the Administrators group**

Members of the Power Users group can install an IBM data server client. Members of the Users group can also install an IBM data server client after they have been enabled to do so. To enable members of the Users group to install an IBM data server client, a member of the Administrators group must ensure the installing user has **write** permission for the following:

- `HKEY_LOCAL_MACHINE\SOFTWARE` registry branch.
- the system directory (for example, `c:\WINNT`).
- the default install path (`c:\Program Files`) or another install path.

Of related interest, a non-administrator can also install fix packs if a non-administrator performed the original installation. However, a non-administrator cannot install fix packs if the original installation was performed by an Administrator user account.

## Command line options to install IBM Data Server Driver Package (Windows)

The IBM Data Server Driver Package can be installed by running the DB2 **setup** command from the command line.

For fix pack images, you can download the driver package that contains the setup program from the IBM Support Fix Central Web site: http://www.ibm.com/support/fixcentral/. Data Server client and driver packages are found under the **Information Management** product group and **IBM Data Server Client Packages** product selection. Select the appropriate installed version and platform and click **Continue**. Click **Continue** again on the next screen and you will be presented with a list of all client and driver packages available for Windows.

The following list describes command-line options available for the **setup** command. For more information about the available Windows Installer options, see http://www.msdn.microsoft.com/.

**/n** [*copy name*]
: Specifies the copy name that you want the installation to use. Specifying this option overrides the installation path that is specified in the response file. If the copy exists, a maintenance installation is performed on that copy. Otherwise, a new installation is performed by using the specified copy name.

**/o**
: Specifies that a new copy installation with a generated default copy name is to be performed.

**/u** [*response file*]
: Specifies the full path and file name of the response file.

**/m**
: Shows the progress dialog during the installation. However, you are not prompted for any input. Use this option with the **/u** option.

**/l** [*log file*]
: Specifies the full path and file name of the log file.

**/p** [*install-directory*]
: Changes the installation path of the product. Specifying this option overrides the installation path that is specified in the response file.

**/i** *language*
: Specifies the two-letter language code of the language in which to perform the installation.

**/?**
: Generates usage information.

After installing IBM Data Server Driver Package, you can optionally create and populate the configuration file, `db2dsdriver.cfg`, with database directory information.

The following are some examples of how to use the command-line parameters:

- To install a new copy with a generated default copy name, use the following command:

```
setup /o
```

- To install a second copy use the following command:
  `setup /n "COPY_NAME"`
- To perform a response file installation, use the following command:
  `setup /u "[Full path to the response file]"`

  A sample response file is located in the `\samples` subdirectory.

# Network installation of IBM Data Server Driver Package (Windows)

You can minimize the effort and disk space required to install IBM Data Server Driver Package on client workstations by installing the code on a network share and registering remote client workstations to use the driver as if it is installed locally. This installation method is available on Windows operating systems only.

To set up a network installation of IBM Data Server Driver Package, you install the code on a code server, rather than on each client workstation, and make the code available to remote client workstations through a network share. You run the **db2dsdpreg** utility to set up each remote client workstation and make the required links to the code server. When a registered remote client initiates a database connection, the driver code is dynamically loaded from the code server as required. The remote client then connects to the database in the typical way. When you update the installed IBM Data Server Driver Package, the updated code is automatically available to the remote client workstations.

The following figure shows a network installation of IBM Data Server Driver Package.

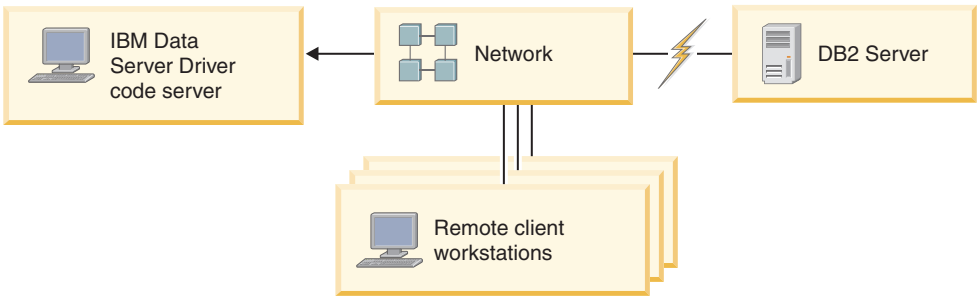



*Figure 2-1. A typical network installation of IBM Data Server Driver Package*

**Note:**

- Remote client workstations that access data on System z® or Power Systems™ data servers must have a DB2 Connect license.
- Remote clients on a 32-bit workstation cannot use a 64-bit code server.
- This installation method requires code to be loaded across a LAN connection. The extent of performance loss at program initialization time depends on variables such as the load on and speed of both the network and the code server.

## Setting up a network installation of IBM Data Server Driver Package (Windows)

To set up a network installation of IBM Data Server Driver Package, you install the driver on the code server, make the code directory available to client workstations, map a network drive from each client workstation to the code server, and register each client workstation.

Perform the following steps to set up a network installation of IBM Data Server Driver Package on a code server and make the code accessible from a remote client workstation.

To set up a network installation:

1. Install IBM Data Server Driver Package on the code server.
2. Make the code directory on the code server available to all remote client workstations.
3. Map a network drive from each remote client workstation to the code server.
4. Register each remote client workstation by running the **db2dsdpreg** utility.

## Making the code directory available to remote client workstations (Windows)

To load the required code from the code server, each of the remote client workstations must be able to read the directory where IBM Data Server Driver Package is installed.

You must install IBM Data Server Driver Package on the code server.

The following procedure uses Windows XP as an example to show how to make the code directory available to all remote client workstations (in read mode).

To make the code directory available to all remote client workstations:

1. On the code server, start Windows Explorer.
2. Select the directory on the code server that will be used to serve client workstations. For this example, select the `C:\Program Files\IBM\IBM DATA SERVER DRIVER` directory to set up the share.
3. Select **File** > **Properties** from the menu bar.
4. Click the **Sharing** tab.
5. Click the **Shared This Folder** radio button.
6. In the **Share Name** field, enter a share name that is eight characters or fewer. For example, enter `DSDRVRSV`.
7. Provide read access to the code directory to all client users:
   a. Click **Permissions**. The Share Permissions window opens.
   b. In the **Group or Users Name** list, highlight the Everyone group.

      **Note:** You can give access to the Everyone group, to a group that you have specifically defined for remote client users, or to individual remote client users.
   c. Select **Read**.
   d. Click **OK** until all windows are closed.

Next, map a network drive from each remote client workstation to the network share on the code server.

## Mapping a network drive from each remote client workstation to the code server (Windows)

Each remote client workstation must have a network drive that is mapped to the network share on the code server.

- Make the code directory on the code server available to all remote client workstations.

- Log on to the remote client workstation as a valid user with shared directory access to the code server.

The following procedure uses Windows XP as an example to show how to map a network drive from the remote client workstation to the network share on the code server.

To map a network drive from the remote client workstation:

1. On the remote client workstation, start Windows Explorer.
2. On the **Tools** menu, click **Map Network Drive**.
3. In the **Drive** list, select the drive to which you want to map the location of the code server.
4. In the **Folder** field, specify the location of the share as follows:

   `\\computer_name\share_name`

   where:

   *computer_name*
   represents the computer name of the code server

   *share_name*
   represents the share name of the shared directory on the code server
5. Select the **Reconnect at Logon** check box to make the share persistent.
6. Click **Finish**.

Next, register each remote client workstation to allow remote clients to use the network installation of IBM Data Server Driver Package as if the code is installed locally.

## Registering remote client workstations by running the db2dsdpreg utility (Windows)

To access a network installation of IBM Data Server Driver Package, remote client workstations must be registered. Run the **db2dsdpreg** utility to register each remote client workstation and create the required links to the code server.

- Map a network drive from the remote client workstation to the code server.
- Ensure that either Microsoft Visual C++ 2005 or the appropriate runtime components of the Visual C++ libraries are installed. The runtime libraries are available from the Microsoft runtime library download Web site. For 32-bit systems, install `vcredist_x86.exe`. For 64-bit systems, install `vcredist_x86.exe` and `vcredist_x64.exe`.
- Log on to the remote client workstation as a valid user with shared directory access to the code server. You must have administrator level privileges on the remote client workstation.

Perform the following steps on each remote client workstation that needs to use the network installation of IBM Data Server Driver Package.

**Restriction:** Remote clients on a 32-bit workstation cannot use a 64-bit code server.

To register a remote client workstation:

1. From a Windows command prompt, issue the following command, where *z* is the network drive that you mapped to the code server:

   `cd z:\bin`

2. Issue the following command to run the **db2dsdpreg** utility and write logging information to a file:

   ```
   db2dsdpreg /l c:\db2dsdpreg.log
   ```

   where `c:\db2dsdpreg.log` is the path where the utility will create the log file. If you do not specify a path for the log file, it is created under `My Documents\DB2LOG` for the current user.

   **Tip:** You can view additional registration options, including options to unregister and reregister, by issuing the following command: `db2dspreg /h`.

3. When the **db2dsdpreg** utility is done, check the messages in the log file (for example, `c:\db2dsdpreg.log`).

   The log file contains error messages for any errors that are encountered when you run the utility.

## Non-DB2 instance merge modules (Windows)

Two types of merge modules are available: DB2 instance merge modules and non-DB2 instance merge modules.

It is recommended that you use non-DB2 instance merge modules. See the related links for details on DB2 instance merge modules.

Using non-DB2 instance Windows Installer merge modules, you can easily add IBM Data Server Driver Package functionality to any product that uses the Windows Installer.

When you merge the modules, you will be prompted to supply the copy name. Multiple copies of IBM Data Server Driver Package products can be installed on the same machine; so each copy is known by its unique name. This name will be used when the installation is performed on each target machine. Choose a name that is unlikely to be already used for another IBM data server driver or DB2 copy. Suitable names include the name of your application, for example, `myapp_dsdrivercopy_1`. If the name is not unique, the installation will fail.

For more information on merge module technology, see the documentation included with your installation authoring product or at http://msdn.microsoft.com.

The following merge module is available for your use:

**`IBM Data Server Driver Package.msm`**
: This module provides support for applications using ODBC, CLI, .NET, OLE DB, PHP, Ruby, JDBC, or SQLJ to access data. It also enables your application to use the IBM Data Server Provider for .NET (DB2 .NET Data Provider and IDS .NET Data Provider). The IBM Data Server Provider .NET is an extension of the ADO.NET interface that enables your .NET applications to quickly and securely access data from DB2 or Informix databases.

DSDRIVER is created by using the merge module described above. Registering IBM Data Server Provider for .NET is based on the version of .NET framework installed on your system. For example, you have to install Microsoft .Net framework 2.0 prior to the installation.

The following merge modules contain language specific messages used by IBM Data Server Driver Package. Depending on the languages of your product, include and install the components in the appropriate merge module.

IBM DSDRIVER Messages - Arabic.msm
IBM DSDRIVER Messages - Bulgarian.msm
IBM DSDRIVER Messages - Chinese(Simplified).msm
IBM DSDRIVER Messages - Chinese(Traditional).msm
IBM DSDRIVER Messages - Croatian.msm
IBM DSDRIVER Messages - Czech.msm
IBM DSDRIVER Messages - Danish.msm
IBM DSDRIVER Messages - Dutch.msm
IBM DSDRIVER Messages - English.msm
IBM DSDRIVER Messages - Finnish.msm
IBM DSDRIVER Messages - French.msm
IBM DSDRIVER Messages - German.msm
IBM DSDRIVER Messages - Greek.msm
IBM DSDRIVER Messages - Hebrew.msm
IBM DSDRIVER Messages - Hungarian.msm
IBM DSDRIVER Messages - Italian.msm
IBM DSDRIVER Messages - Japanese.msm
IBM DSDRIVER Messages - Korean.msm
IBM DSDRIVER Messages - Norwegian.msm
IBM DSDRIVER Messages - Polish.msm
IBM DSDRIVER Messages - Portuguese(Brazilian).msm
IBM DSDRIVER Messages - Portuguese(Standard).msm
IBM DSDRIVER Messages - Romanian.msm
IBM DSDRIVER Messages - Russian.msm
IBM DSDRIVER Messages - Slovak.msm
IBM DSDRIVER Messages - Slovenian.msm
IBM DSDRIVER Messages - Spanish.msm
IBM DSDRIVER Messages - Swedish.msm

## Validating IBM Data Server Driver Package (Windows) installation

You can set up the IBM Data Server Driver Package on the Windows platform. Starting with DB2 Version 9.7 Fix Pack 3a, follow the instructions to validate your installation of the IBM Data Server Driver (ds driver) for the most popular interfaces, such as ODBC, ADO.NET and Visual Studio.

You can download the IBM Data Server Driver Package using the following link: https://www-304.ibm.com/support/docview.wss?rs=4020&uid=swg27016878.

After installing IBM Data Server Driver Package, you can validate the installation for CLI / ODBC using the **db2cli validate** command.

Use the following steps to validate your installation:

1. Create an alias in `db2dsdriver.cfg` file. You can then populate the configuration file with dsn alias name, database name, hostname, and the port to which connection is to be made. Starting from DB2 Version 9.7 Fix Pack 4, the `db2dsdriver.cfg` path for IBM Data Server Driver Package on Windows can be obtained by running **db2cli validate**, to show the `db2dsdriver.cfg` file location.
2. Validate the dsn alias with db2cli.
3. Create an odbc dsn to the alias.
4. Test the odbc dsn using MS Excel.

## db2dsdriver.cfg

For Example, below is a sample db2dsdriver.cfg file configured with dsn alias as **sampledsn**, database name as **sample**, hostname as **samplehost.domain.com** and port as **19766**.

```
<configuration>
  <dsncollection>
   <dsn alias="sampledsn" name="sample" host="samplehost.domain.com" port="19766"/>
  </dsncollection>
  <databases>
   <database name="sample" host="samplehost.domain.com" port="19766">
   </database>
  </databases>
</configuration>
```

## Location of the db2dsdriver.cfg file

The location of db2dsdriver.cfg file differs between IBM Data Server Client and IBM Data Server Driver.

- In IBM Data Server Client: *%instance_path%/cfg*
- In IBM Data Server Server: *%installation_path%/cfg*

You can find the location of the db2dsdriver.cfg file using testconn20.exe or testconn40.exe utility with a dummy argument. The following example shows the usage of this dummy argument:

```
C:\Program Files\IBM\IBM DATA SERVER DRIVER\bin>testconn40.exe dummy

Step 1: Printing version info
        .NET Framework version: 4.0.30319.1
        64-bit
        DB2 .NET provider version: 9.7.4.4
        DB2 .NET file version: 9.7.4.4
        Capability bits: ALLDEFINED
        Build: 20110303
        Factory for invariant name IBM.Data.DB2 verified
        Factory for invariant name IBM.Data.Informix verified
        IBM.Data.Informix from DbFactory is the Common Informix .NET provider
IBM Database Add-ins is not installed properly
        Elapsed: 1.875024

Step 2: Validating db2dsdriver.cfg against db2dsdriver.xsd schema file
        C:\PROGRA~1\IBM\IBMDAT~1\cfg\db2dsdriver.cfg against
C:\Program Files\IBM\IBM DATA SERVER DRIVER\cfg\db2dsdriver.xsd
        Elapsed: 0.0156252
.. .. .. .. .. .. .. ..
.. .. .. .. .. .. .. ..
```

## Validating dsn alias

You can run `db2cli validate -dsn sampledsn` to validate the dsn alias "sampledsn" configured in the **db2dsdriver.cfg** file. If the entries are correct, validation will be successful with the following output.

```
C:\Program Files\IBM\IBM DATA SERVER DRIVER\bin>db2cli validate -dsn
sampledsn
IBM DATABASE 2 Interactive CLI Sample Program
(C) COPYRIGHT International Business Machines Corp. 1993,1996
All Rights Reserved
Licensed Materials - Property of IBM
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
```

```
Header :
  --------------------------------------------------------------------------
  [ CLI Driver Version   : 09.07.0000 ]
  [ Informational Tokens : "DB2 v9.7.301.364", "s101112",IP23228","Fixpack  3" ]

  [ CLI Driver Type      : IBM Data Server Driver For ODBC and CLI ]
  --------------------------------------------------------------------------


db2dsdriver.cfg Schema Validation :
Success: The schema validation operation completed successfully.
The configuration file named db2dsdriver.cfg is valid.

Warning: The schema validation operation completed successfully.
The following data source name was not found in the db2cli.ini
file: "sampledsn".

db2dsdriver.cfg Validation :
  --------------------------------------------------------------------------
  [ DB2DSDRIVER_CFG_PATH env var : unset ]
  [ db2dsdriver.cfg Path         : C:\ProgramData\IBM\DB2\IBMDBCL1\cfg\db2dsdriv
er.cfg ]
  --------------------------------------------------------------------------
  [ Keywords used by CLI for DSN : sampledsn ]
    Keyword                                    Value
    ------------------------------------------------
   DATABASE                                   sample
   HOSTNAME                                   samplehost.domain.com
   PORT                                       19766
  --------------------------------------------------------------------------

The validation completed.
```

If you are a DB2 Connect customer but are not using the server based license key or a DB2 Connect server, you may get this message from your connection:

```
[IBM][CLI Driver] SQL1598N  An attempt to connect to the database server
failed because of a licensing problem.  SQLSTATE=42968
```

If you are using DB2 Connect Unlimited Edition for z/OS, the recommend solution is to use a server based license key. This one step will prevent the need for client based license keys. For details, see the topic about activating the license key for DB2 Connect Unlimited Edition for System z.

If you are unable to use the above solution, please take the DB2 Connect license key from the DB2 Connect Edition you have purchased (eg: db2conpe.lic) and place it in the `C:\Program Files\IBM\IBM DATA SERVER DRIVER\license` license directory, underneath the installation location for the IBM Data Server Driver Package.

Once **db2dsdriver.cfg** is populated with the correct database connection information, register the dsn alias with the ODBC driver manager as a datasource. On Windows operating systems, you can make the data source available to all users of the system (a system data source), or only to the current user (an user data source).

## Validating CLPPlus

To verify that CLPPlus works properly, you can connect to the dsn alias sampledsn defined in the `db2dsdriver.cfg` file. Follow these steps:

1. At the operating system prompt, use the **clpplus** command with **username** and **dsn-alias** parameters to start CLPPlus.
2. When prompted, enter the password associated with the username you provided.

A successful connection indicates that CLPPlus works properly. The following example output shows the two-step verification and successful connection:

```
C:\>clpplus db2admin@sampledsn
CLPPlus: Version 1.4
Copyright (c) 2009, 2011, IBM CORPORATION.  All rights reserved.

Enter password: **********

Database Connection Information :
---------------------------------
Hostname = samplehost.domain.com
Database server = DB2/NT  SQL09074
SQL authorization ID = db2admin
Local database alias = SAMPLEDSN
Port = 19766
```

## Creating odbc dsn to the alias

Create a system dsn for dsn alias **sampledsn** using the following command:

```
db2cli registerdsn -add sampledsn -system
```

Starting from DB2 Version 9.7 Fix Pack 4, use the binary **db2cli32** command, instead of **db2cli**, if you are using a 32-bit driver along with the 64-bit installer in a 64-bit Windows machine. For example:

```
db2cli32 registerdsn -add sampledsn -system
```

## Testing odbc dsn using MS Excel

You can test the newly created ODBC dsn using Microsoft applications like MS Excel.

**Procedure**

1. Launch odbc administrator tool from `Control Panel->Administrative Tools-> Data Sources (ODBC) for 64 bit binary`. For 32-bit binary in a 64-bit machine, launch odbc administrator tool from `System Drive:\windows\SysWOW64\ odbcad32.exe`. Example: `c:\windows\syswow64\ odbcad32.exe`
2. The list of user data sources is displayed, by default. Click **System DSN** tab.
3. Select the newly created dsn (sampledsn in our example) and click **Configure**.
4. Provide user name, password and click **Connect**. The message "Connection tested successfully" is displayed.
5. Now launch Microsoft applications like MS Excel and use the newly created dsn. To do the same, bring up MS excel. For example,
   a. Launch MS Excel 2003.
   b. Go to `Data->Import External Data-> New Database Query`. The list of odbc dsn's is displayed in a list box.

      **Note:** A 32-bit excel application will show only 32-bit dsn's in the list box and 64-bit excel will show only 64-bit dsn's in the list box.
   c. Select the dsn that (sampledsn in our example) you want to connect to, and provide the login details.

d. The list of tables is displayed in the database.

## Testing connectivity using ADO.NET

You can verify that DB2 ADO.NET drivers are installed correctly and are fully operational by running the **testconn20.exe** utility. Use the **-dtc** command option to verify XA transaction support setup. To verify the runtime build with .NET Framework 4.0, you can use testconn40.exe.

```
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: 2.0.50727.3615
        64-bit
        DB2 .NET provider version: 9.0.0.2
        DB2 .NET file version: 9.7.3.2
        Capability bits: ALLDEFINED
        Build: 20101113
        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 09.07.0003
        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.
```

The error message `VSAI is not installed properly` can be ignored. This error shows up because VSAI is only available in 32 bit, and is failed to be detected by a 64 bit **testconn20**. The 32 bit version of **testconn20** should properly report VSAI information.

If you want to test connectivity to a particular server without adding an alias to the `db2dsdriver.cfg`, you can specify full connectivity information in the connection string.

```
C:\Program Files\IBM\IBM DATA SERVER DRIVER\bin>testconn20 -dtc
"database=sample;server=samplehost.domain.com:19766;uid=username;pwd=password"
```

If you want to test connectivity for your 32 bit applications that are running in a 64 bit environment you can use the 32-bit version of the **testconn20** utility.

```
C:\Program Files\IBM\IBM DATA SERVER DRIVER\bin>testconn20_32 -dtc
"database=sampledsn;uid=username;pwd=password" adding MSDTC step

Step 1: Printing version info
        .NET Framework version: 2.0.50727.3615
        DB2 .NET provider version: 9.0.0.2
        DB2 .NET file version: 9.7.3.2
        Capability bits: ALLDEFINED
        Build: 20101113
        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 assembly version: 9.1.0.0
        VSAI file version: 9.7.3.1012
        Elapsed: 1.0000192

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 09.07.0003
        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.
```

## Creating connection in Server Explorer using IBM Database Add-ins for Visual Studio

After installing IBM Database Add-ins for Visual Studio, you can create a connection in Server explorer.

To create a connection:

1. Launch Visual Studio.
2. Right click on the Data Connections node in server explorer and click "Add Connection..."
3. In the Add Connection dialog, select IBM DB2 and IDS Data Provider for .Net Data source.
4. Click on the database drop down and the aliases defined in the `dsdriver.cfg` file will be listed. You can either pick the alias from the drop down or type the alias name in the database name field.
5. Select `sampledsn`, enter the username and password and click on Test Connection. A message box will appear to indicate that the test connection passed. Close the message box.
6. Click on the OK button in the connection dialog. The connection is now created in the server explorer.

## Validating using testconn utility

Testconn utility (testconn20.exe and testconn40.exe) is located in the `sqllib/bin` subdirectory of the home directory of the database instance owner. Testconn20.exe is a .NET Framework 2.0 executable utility and testconn40.exe is a .NET Framework 4.0 executable utility.

### Command syntax


### Command parameters

**dtc**

Validates a connection for distributed transaction.

**ids**

Validates using the Common IDS .NET provider.

**idsold**

Validate using IBM Informix Client SDK .NET provider.

**validateExtendedInsight / validateEI**

Validates the db2dsdriver.cfg file setup and OPM EI monitoring configuration for the database mentioned in the connection string.

**validateExtendedInsightwithConnect / validateEIwithConnect**

In addition to validating the db2dsdriver.cfg file setup and OPM EI monitoring configuration for the database mentioned in the connection

string, this switch makes a connection to the database and displays whether monitoring is enabled for this database.

**maxSteps**

Runs only for the number of steps mentioned.

**specific**

validates using the fixpack specific provider that exists under `netf20/specific` and `netf40/specific` directories.

**enumerateremotedbs / enumremotedbs**

Lists the databases available on the remote server indicated in the connection string.

**asp**

Suggests items for consideration when working with ASP .NET applications.

**discover**

Lists the DB2 servers that are accessible via DB2 administration server (DAS).

**enumeratedbs / enumdbs**

Enumerates the databases accessible from the machine.

**enumeratelocaldbs / enumlocaldbs**

Lists the databases available on the local machine.

## Installing IBM Data Server Driver Package (Linux and UNIX)

On Linux and UNIX operating systems, IBM Data Server Driver Package simplifies application deployment. This driver, which has a small footprint, is designed to be redistributed by independent software vendors (ISVs) and to be used for application distribution in mass deployment scenarios typical of large enterprises.

To install IBM Data Server Driver Package, you need to obtain the compressed file that contains this driver package. Download this file from the IBM Support Fix Central Web site: http://www.ibm.com/support/fixcentral/. Data Server clients and drivers are found under the **Information Management** product group and **IBM Data Server Client Packages** product selection. Select the appropriate installed version and platform and click **Continue**. Click **Continue** again on the next screen and you will be presented with a list of all client and driver packages available for your platform.

On Linux and UNIX operating systems, IBM Data Server Driver Package is installed by running the **installDSDriver** command. This driver package includes database drivers for Java, ODBC/CLI, PHP, and Ruby on Rails, each of which is stored in its own subdirectory. The Java and ODBC/CLI drivers are compressed.

The basic extraction steps for each driver are the same:

1. Uncompress the Data Server Driver Package archive.
2. Copy the files onto the target machine.
3. For the Java and ODBC/CLI drivers, uncompress the driver file into your chosen install directory on the target machine.
4. Optional: remove the compressed driver file.

**Java**

Path: `jdbc_sqlj_driver/`*platform*

Filename: `db2_db2driver_for_jdbc_sqlj.zip`

For installation instructions, see: Installing the IBM Data Server Driver for JDBC and SQLJ.

**ODBC/CLI**

Path: `odbc_cli_driver/`*platform*

Filename: `ibm_data_server_driver_for_odbc_cli.tar.Z`

For installation instructions, see: Installing the IBM Data Server Driver for ODBC and CLI.

**PHP**

Path: `php_driver/`*platform*`/php32` or `php_driver/`*platform*`/php64`

Files: `ibm_db2_`*n*`.`*n*`.`*n*`.so`, `ibm_db2_`*n*`.`*n*`.`*n*`.so`, `pdo_ibm_`*n*`.`*n*`.`*n*`.so`, where *n* represents the version of the extension.

Prerequisite: The PHP drivers require the ODBC/CLI driver also included in this driver package to be installed.

For installation instructions, see: Setting up the PHP environment on Linux or UNIX.

**Ruby on Rails**

Path: `ruby_driver/`*platform*

File: `ibm_db-0.10.0.gem`

Prerequisite: The Ruby on Rails driver requires the ODBC/CLI driver also included in this driver package to be installed.

To install the Ruby on Rails driver, from the location of the gem file, run the following command: **gem install ibm_db-0.10.0.gem**. To validate the installation, see: Verifying installation with IBM Data Servers on Rails applications.

1 After installing Data Server Driver Package, you can optionally create and populate the configuration file, `db2dsdriver.cfg`, with database directory information.

## Installation methods for IBM data server clients

Common and alternative methods for installing an IBM data server client or driver.

Clients are commonly installed on machines where there is no server present. You do not need to install a client if you already installed a server product is already installed because the server includes all the functionality present in an IBM data server client.

### Common situations

The common method for installing an IBM data server client or driver is to run the installation program provided on a product DVD (**setup** command on Windows operating systems and **db2setup** command on Linux and UNIX operating systems). The IBM Data Server Client installation image is included in the database server installation image.

### Automation of large-scale deployments

One group of methods automates the deployment of large numbers of clients:

- **Response file.** You can automate client installation by using the response file installation method. A response file installation lets you install database products without any user interaction.
- **Third-party deployment tools.** You can install clients using deployment tools or methods such as Windows Active Directory, Windows Systems Management Server (SMS), or Tivoli® products.

### Use of Windows operating system capabilities

Another group of options uses Windows operating system capabilities:

- **Windows thin client topology.** This option is supported for the IBM Data Server Client and DB2 Connect Personal Edition. A thin client topology is where client code is installed in a shared Windows directory on a single code server rather than on the local hard disk of each client workstation. Individual client workstations connect to the shared Windows directory on the code server to run the Data Server Client code.
- **A Windows non-administrator ID.** The common installation method uses a Windows administrator user ID: that is, a user ID in the Administrators group. However, you can also install an IBM data server client using a user ID that is part of the Windows Power Users group or Users group. This method is suitable when the user ID performing the installation does not have administrator privileges. The DB2 product also supports the Windows Elevated Privileges mechanism. It is recommended to use Windows Elevated Privileges to allow a non-administrator to install an IBM data server client.

### Linux and UNIX alternatives

On Linux and UNIX operating systems, an alternative installation method provided for database servers is also applicable to clients: the **db2_install** script.

Also, IBM Data Server Driver for ODBC and CLI is available as a tar file.

### Separate client instances

If a database server product is installed, you can use a separate client instance instead of using a server instance that also serves as a client instance.

To create a separate client instance, use the **db2icrt** command with the **-s** option, as shown in the following example:

```
db2icrt -s client instname
```

## Installing IBM data server clients (Windows)

Instructions to install any IBM data server client type, namely the IBM Data Server Client, the IBM Data Server Runtime Client, and the IBM Data Server Driver Package. The main procedure covers a simple, but common, case where there is no DB2 database product already installed.

If the machine already has a prior version of a client installed, you should first review topics that cover upgrading.

If the machine already has a DB2 database server product installed, it is not necessary to install a client because the DB2 database server provides all the capability found in an IBM data server client.

Before installing IBM data server clients or client package:

- You have determined which client best suits your need.
- You have located a DVD or other install image that you need. You can download an image from the IBM Support Fix Central Web site: http://www.ibm.com/support/fixcentral/. Data Server client and driver packages are found under the **Information Management** product group and **IBM Data Server Client Packages** product selection. Select the appropriate installed version and platform and click **Continue**. Click **Continue** again on the next screen and you will be presented with a list of all client and driver packages available for Windows. Ensure you have the appropriate 32-bit or 64-bit version, depending on your machine.
- You have a Windows user account that is part of the Administrators group.

  **Note:** If a non-Administrator user account is going to do the product installation, then the VS2005 runtime library must be installed before attempting to install a DB2 database product. The VS2005 runtime library is needed on the operating system before the DB2 database product can be installed. The VS2005 runtime library is available from the Microsoft runtime library download web site. There are two choices: choose `vcredist_x86.exe` for 32-bit systems or `vcredist_x64.exe` for 64-bit systems.
- Your system meets all memory, disk space, and installation requirements. The installation program will check disk space and basis system requirements, and notify you if there is a problem.

**Restrictions**

- No other DB2 database product can be installed in the same path if one of the following products is already installed:
  - IBM Data Server Runtime Client
  - IBM Data Server Driver Package
  - *DB2 Information Center*
- The DB2 Setup wizard fields do not accept non-English characters.

This procedure covers the simple case. Information for other cases is covered elsewhere in this topic.

To install any IBM data server client on Windows:

1. Log on to the system with the user account that you want to use to perform the installation.
2. Optional: Shut down any other programs.
3. Insert the DVD into the drive. The autorun feature starts the DB2 Setup wizard which determines the system language and starts the setup program for that language.

   For the Data Server Client, you can run the DB2 Setup wizard in a language other than the default system language by manually invoking the DB2 Setup wizard and specifying a language code. For example, the **setup -i fr** command runs the DB2 Setup wizard in French. For the Data Server Runtime Client or Data Server Driver Package, there are separate install images for each language.
4. To install an IBM data server client or client package:

- If you are installing a Data Server Client, launch the DB2 Setup wizard, when the DB2 Launchpad opens, choose **Install a Product**. Follow the DB2 Setup wizard's prompts.
- If you are installing a Data Server Runtime Client, it does not have a launchpad. See the Related Links for **setup** command parameters.

  If you are installing a second copy of the Data Server Runtime Client, the command is:

  ```
  setup /v" TRANSFORMS=:InstanceId1.mst MSINEWINSTANCE=1"
  ```

  To install each subsequent copy of the Data Server Runtime Client (up to a maximum of 16 copies), modify the command by incrementing `InstanceId`*n*, for example:

  ```
  setup /v" TRANSFORMS=:InstanceId2.mst MSINEWINSTANCE=1"
  ```

  **Note:** It is strongly recommended that installing multiple copies is for advanced users.
- If you are installing Data Server Driver Package, run the **setup** command from the product DVD, or install the driver from a fix pack image by downloading the driver from http://www.ibm.com/support/docview.wss?rs=71&uid=swg27007053. If you are installing a fix pack image, see the Related Links for the installation command options of the **setup** command.

  If you are installing a second copy of Data Server Driver Package, the following methods can be used:
  - To perform a new copy installation with a generated default copy name:

    ```
    setup /o
    ```
  - If the copy name already exists, perform a maintenance (or upgrade) installation on that copy. Otherwise, perform the new installation by using the specified copy name.

    ```
    setup /n copyname
    ```

  After installing Data Server Driver Package, you can optionally create and populate the configuration file, `db2dsdriver.cfg`, with database directory information.

5. If you are installing a Data Server Client on a machine that already has a DB2 Universal Database (UDB) Version 8 copy installed, you will be presented with the option to install a new copy or to upgrade the DB2 UDB Version 8 copy.

   Installing a new copy preserves the DB2 UDB Version 8 copy and installs an additional DB2 Version 9 copy. Choosing to upgrade will copy the DB2 UDB Version 8 client instance settings to the DB2 Version 9 copy then remove the DB2 UDB Version 8 copy.

   **Note:** If a machine already has a DB2 UDB Version 8 copy installed, the Version 9 copies cannot be set to default.

   If you are installing a Data Server Runtime Client, the installation program always installs a new copy. To upgrade a DB2 UDB Version 8 client instance as a subsequent step, see topics on migration.

After completing this procedure, the product is now installed at the location you specified during the installation. As part of the IBM Data Server Client installation procedure, an instance of the DB2 database manager is created. The instance is called "DB2" if there is no other instance called "DB2". If you already have a copy of DB2 Version 8 or DB2 Version 9.1 installed, the default instance is DB2_01.

The default installation path of Data Server Client and Data Server Runtime Client is `Program Files\IBM\sqllib`. If a second copy is installed in the same machine, the default directory name is `Program Files\IBM\sqllib_01`. In general, the default directory name is `sqllib_nn` where *nn* is the number of copies installed in that machine minus one.

The default installation path of Data Server Driver Package is `Program Files\IBM\IBM DATA SERVER DRIVER`. If a second copy is installed in the same machine, the default directory name is `Program Files\IBM\IBM DATA SERVER DRIVER_02`. In general, the default directory name is `IBM DATA SERVER DRIVER_nn` where *nn* is the generated number to make this directory unique.

If you want to install more than one copy of the Data Server Driver Package, you can have a maximum of 16 copies. Each copy must be installed to different directories.

The default copy name of the Data Server Client or Data Server Runtime Client is `DB2COPY1`

The default copy name of the Data Server Driver Package is `IBMDBCL1`

This installation does not include product documentation. See the related links for options for installing or accessing the *DB2 Information Center*.

After installing your IBM data server client, the next step is to configure it to access remote DB2 database servers.

**Notes on installing using a user account that is not a member of the Administrators group**

Members of the Power Users group can install an IBM data server client. Members of the Users group can also install an IBM data server client after they have been enabled to do so. To enable members of the Users group to install an IBM data server client, a member of the Administrators group must ensure the installing user has **write** permission for the following:

- `HKEY_LOCAL_MACHINE\SOFTWARE` registry branch.
- the system directory (for example, `c:\WINNT`).
- the default install path (`c:\Program Files`) or another install path.

Of related interest, a non-administrator can also install fix packs if a non-administrator performed the original installation. However, a non-administrator cannot install fix packs if the original installation was performed by an Administrator user account.

## Non-DB2 instance merge modules (Windows)

Two types of merge modules are available: DB2 instance merge modules and non-DB2 instance merge modules.

It is recommended that you use non-DB2 instance merge modules. See the related links for details on DB2 instance merge modules.

Using non-DB2 instance Windows Installer merge modules, you can easily add IBM Data Server Driver Package functionality to any product that uses the Windows Installer.

When you merge the modules, you will be prompted to supply the copy name. Multiple copies of IBM Data Server Driver Package products can be installed on the same machine; so each copy is known by its unique name. This name will be used when the installation is performed on each target machine. Choose a name that is unlikely to be already used for another IBM data server driver or DB2 copy. Suitable names include the name of your application, for example, `myapp_dsdrivercopy_1`. If the name is not unique, the installation will fail.

For more information on merge module technology, see the documentation included with your installation authoring product or at http://msdn.microsoft.com.

The following merge module is available for your use:

**IBM Data Server Driver Package.msm**
: This module provides support for applications using ODBC, CLI, .NET, OLE DB, PHP, Ruby, JDBC, or SQLJ to access data. It also enables your application to use the IBM Data Server Provider for .NET (DB2 .NET Data Provider and IDS .NET Data Provider). The IBM Data Server Provider .NET is an extension of the ADO.NET interface that enables your .NET applications to quickly and securely access data from DB2 or Informix databases.

DSDRIVER is created by using the merge module described above. Registering IBM Data Server Provider for .NET is based on the version of .NET framework installed on your system. For example, you have to install Microsoft .Net framework 2.0 prior to the installation.

The following merge modules contain language specific messages used by IBM Data Server Driver Package. Depending on the languages of your product, include and install the components in the appropriate merge module.

IBM DSDRIVER Messages - Arabic.msm
IBM DSDRIVER Messages - Bulgarian.msm
IBM DSDRIVER Messages - Chinese(Simplified).msm
IBM DSDRIVER Messages - Chinese(Traditional).msm
IBM DSDRIVER Messages - Croatian.msm
IBM DSDRIVER Messages - Czech.msm
IBM DSDRIVER Messages - Danish.msm
IBM DSDRIVER Messages - Dutch.msm
IBM DSDRIVER Messages - English.msm
IBM DSDRIVER Messages - Finnish.msm
IBM DSDRIVER Messages - French.msm
IBM DSDRIVER Messages - German.msm
IBM DSDRIVER Messages - Greek.msm
IBM DSDRIVER Messages - Hebrew.msm
IBM DSDRIVER Messages - Hungarian.msm
IBM DSDRIVER Messages - Italian.msm
IBM DSDRIVER Messages - Japanese.msm
IBM DSDRIVER Messages - Korean.msm
IBM DSDRIVER Messages - Norwegian.msm
IBM DSDRIVER Messages - Polish.msm
IBM DSDRIVER Messages - Portuguese(Brazilian).msm
IBM DSDRIVER Messages - Portuguese(Standard).msm
IBM DSDRIVER Messages - Romanian.msm
IBM DSDRIVER Messages - Russian.msm

IBM DSDRIVER Messages - Slovak.msm
IBM DSDRIVER Messages - Slovenian.msm
IBM DSDRIVER Messages - Spanish.msm
IBM DSDRIVER Messages - Swedish.msm

## Installing IBM data server clients (Linux and UNIX)

This task describes how to install an IBM data server client on Linux or UNIX. The instructions apply to the IBM Data Server Client and the IBM Data Server Runtime Client. The main procedure covers a simple, but common, case where no DB2 database product is already installed.

If the machine already has a prior version of a client installed, you should first review topics that cover upgrading.

If the machine already has a DB2 database server product installed, it is not necessary to install a client because the DB2 database server provides all the capability found in IBM Data Server Client.

- You have determined which client best suits your needs: the Data Server Client or the Data Server Runtime Client.
- You have located a DVD or other install image that you need. You can download an image from the IBM Support Fix Central Web site: http://www.ibm.com/support/fixcentral/. Data Server clients and drivers are found under the **Information Management** product group and **IBM Data Server Client Packages** product selection. Select the appropriate installed version and platform and click **Continue**. Click **Continue** again on the next screen and you will be presented with a list of all client and driver packages available for your platform.
- Your system meets all memory, disk space, and installation requirements. The installation program will check disk space and basis system requirements, and notify you if there is a problem.
- Installing an IBM data server client on the Solaris operating system or on HP-UX requires that you update your kernel configuration parameters. This is also recommended for Linux.

To install any IBM data server client on Linux or UNIX:

1. Insert and mount the appropriate DVD.
2. Change to the directory where the DVD is mounted.
3. Enter the `./db2setup` command to start the DB2 Setup wizard.
4. Choose **Install a Product** when the DB2 Launchpad opens.
5. Select the client you want to install.
6. Follow the DB2 Setup wizard's prompts. Help is available in the wizard to guide you through the remaining steps.

When installation is complete, the IBM data server client is installed by default in the following directories:

**Linux** `/opt/ibm/db2/V9.7`

**UNIX** `/opt/IBM/db2/V9.7`

This installation does not include product documentation. See the related links for options for installing or accessing the *DB2 Information Center*.

After installing your IBM data server client, the next step is to configure it to access a remote DB2 server.

**Notes on national language installations**

You can run the DB2 Setup wizard in a language other than the default system language by manually invoking the DB2 Setup wizard and specifying a language code. For example, the **./db2setup -i fr** command runs the DB2 Setup wizard in French. However, the DB2 Setup wizard fields do not accept non-English characters.

**Notes on installing on a machine that has an existing DB2 Version 9.5 client**

The default directory name for the first copy is `V9.7`. If a copy is already installed, the second installation shows a default directory name of `V9.7_01`. In general, the default directory name is `V9.7_`*nn* where *nn* refers to the number of copies installed minus one.

**Notes on installing on a machine that has an existing pre-DB2 Version 9.5 client**

Installing a Data Server Client or Data Server Runtime Client on a system that already has either a DB2 Universal Database (UDB) Version 8 or DB2 Version 9 client preserves previous copy and installs an additional DB2 Version 9.5 (or higher) copy. For information on upgrading client instances to DB2 Version 9.5 (or higher), see the upgrading topics.

## IBM Data Server Runtime Client installation command-line options

You can install IBM Data Server Runtime Client using the **db2setup.exe** command on Linux or UNIX operating systems or the **setup.exe** command on Windows operating systems. The parameters of the two commands are different.

The following list describes selected popular standard Windows Installer command line options available when you run **setup.exe** to install IBM Data Server Runtime Client on Windows operating systems. For more information on the available Windows Installer options, see http://www.msdn.microsoft.com/.

**/l*v[***log file name***]**
: This option allows you to create a log of the installation. You can use the log to troubleshoot any problems that you encountered during the installation. If you specify a directory, this directory must exist or must be created prior installation. Otherwise, the installation returns an error and fails.

**/v**
: This option allows you to pass additional command line options and public properties to the Windows Installer. You must specify this option to perform a response file installation.

**/qn**
: This option allows you to perform a silent installation with no User Interface (UI), except for a window that the Windows installer displays while it extracts files from the installation package before it starts the actual installation.

**/qb!**
: This option displays a basic user interface which shows simple progress and error message handling and hides the **Cancel** button, except in a window that the Windows installer displays while it extracts files from the installation package before it starts the actual installation.

**/s** This option allows you to suppress the initialization dialog for the installation.

**/L** This option allow you to change the setup language by specifying the language identifier. For example, to specify French as the setup language, specify the French language identifier, **setup.exe /L1036** command.

*Table 2-6. Language Identifiers*

| Language | Identifier |
|---|---|
| Arabic (Saudi Arabia) | 1025 |
| Bulgarian | 1026 |
| Chinese (Simplified) | 2052 |
| Chinese (Traditional) | 1028 |
| Croatian | 1050 |
| Czech | 1029 |
| Danish | 1030 |
| Dutch (Standard) | 1043 |
| English | 1033 |
| Finnish | 1035 |
| French (Standard) | 1036 |
| German | 1031 |
| Greek | 1032 |
| Hebrew | 1037 |
| Hungarian | 1038 |
| Italian (Standard) | 1040 |
| Japanese | 1041 |
| Korean | 1042 |
| Norwegian (Bokmal) | 1044 |
| Polish | 1045 |
| Portuguese (Brazilian) | 1046 |
| Portuguese (Standard) | 2070 |
| Romanian | 1048 |
| Russian | 1049 |
| Slovak | 1051 |
| Slovenian | 1060 |
| Spanish (Traditional Sort) | 1034 |
| Swedish | 1053 |
| Turkish | 1055 |

Here are the public properties that you can specify to control the installation of Data Server Runtime Client:

- These parameters must be the last parameters in the command line.
- **RSP_FILE_PATH** - This contains the full path to the response file that you use to install Data Server Runtime Client. This is valid only when you specify **/qn**.

To perform a response file installation with a log file, the following command line parameter should be used:

```
   setup /v" /l*v [Full path to a log file] /qn
RSP_FILE_PATH=[Full path to the response file]"
```

The example assumes that no copy of the client is already installed. If one or more copies exist, the command is different. To install a second copy using a response file, use the following command:

```
   setup /v" TRANSFORMS=:InstanceId1.mst MSINEWINSTANCE=1
         /qn RSP_FILE_PATH=[Full path to the response file]"
```

### Note

- On Windows operating systems, the **setup.exe** file in a DB2 fix pack release follows the format <*release*><*fix pack*><*platform*><*product*>.exe. For example, the **setup** file **v9.7fp3_nt32_rtcl.exe**, where *release*=v9.7, *fix pack*= fp3, *platform*=nt32 and *product*=rtcl, is used when using the IBM Data Server Runtime Client with DB2 Version 9.7 Fix Pack 3 on a NT32 platform.

## Uninstalling an IBM data server client

This topic describes how to uninstall an IBM data server client.

Perform one of the following steps to uninstall an IBM data server client:

- To remove an IBM data server client from a Linux or UNIX operating system, run the **db2_deinstall -a** command from the *DB2DIR*/install directory, where *DB2DIR* is the location that you specified when you installed the data server client.
- To remove an IBM data server client from a Windows operating system, use the Add/Remove Programs window, accessible through the Windows Control Panel. Refer to your operating system's help for more information about removing software products from your Windows operating system.

  **Note:** On Windows, the Add/Remove Programs window can be used to uninstall any of the IBM data server clients. If you are removing an IBM Data Server Client, you can run the **db2unins** command. However, this method cannot be used to remove either the IBM Data Server Runtime Client or the IBM Data Server Driver Package. For more information, see the **db2unins** command topic.

# Chapter 3. Programming applications to use the IBM Data Server 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.Informix");
DbConnection conn = factory.CreateConnection();
DbConnectionStringBuilder sb = factory.CreateConnectionStringBuilder();

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

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 example above, the `"IBM.Data.Informix"` 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 familyIDS databases, just as a `IfxConnection` 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 above 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 stores_demo 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 `IfxConnection` class.

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=stores_demo";
// When used, attempts to connect to the stores_demo 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 `dsdriver.cfg` file.

Example 2:

```
String cs = "Server=srv:9089;Database=stores_demo;UID=informix;PWD=ab1d";
// When used, attempts to connect to the stores_demo database on the server
// 'srv' through port 9089 using 'informix' and 'ab1d' as the user id and
// password respectively.
```

**Note:** By default Informix is installed with DRDA support and the port number is 9089. However, if you install Informix without DRDA support the default port number is 9088.

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

     ```
     String connectString = "Database=stores_demo";
     IfxConnection conn = new IfxConnection(connectString);
     conn.Open();
     return conn;
     ```

   - Connecting to a database in Visual Basic .NET:

     ```
     Dim connectString As String = "Database=stores_demo"
     Dim conn As IfxConnection = new IfxConnection(connectString)
     conn.Open()
     Return conn
     ```

3. Use the `IfxConnection` 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 Informix 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 connection string keywords for the following:

- The minimum and maximum pool size (min pool size, max pool size)

- The length of time a connection can be idle before it is returned to the pool (connection lifetime)

## SQL data type representation in ADO.NET database applications

The parameters for SQL data types can be sent through or received into either a .NET Framework data type, or an IBM Data Server Provider for .NET data type.

- The advantage of using IBM Data Server Provider for .NET data types is that the provider can fit all the values supported by IDS. Using the .NET Framework data types could result in data truncation errors for some values.
- The advantages of using .NET Framework data types is better integration with rest of the .NET Framework classes, which the IBM Data Server Provider for .NET data types would lack.

The following table shows mappings between IfxType data types, Informix data types, Microsoft .NET Framework types, and IfxTypes classes and structures.

| **Category** | **IfxType Data Type** | **Informix Data Type** | **.NET Framework Data Type** | **IfxTypes Namespace Classes and Structures** |
|---|---|---|---|---|
| Character data | Char | CHAR | String | IfxString |
| | VarChar | VARCHAR | String | IfxString |
| | LongVarChar | LVARCHAR | String | IfxString |
| Date/Time data | Date | DATETIME (date precision) | DateTime | IfxDate |
| | Time | DATETIME (time precision) | TimeSpan | IfxTime |
| | Datetime | DATETIME (date and time precision) | DateTime | IfxTimeStamp |
| LOB data | Blob | BLOB, BYTE | Byte[] | IfxBlob |
| | Clob | CLOB, TEXT | String | IfxClob |

| Category | IfxType Data Type | Informix Data Type | .NET Framework Data Type | IfxTypes Namespace Classes and Structures |
|---|---|---|---|---|
| Numeric data | SmallInt | BOOLEAN, SMALLINT | Int16 | IfxInt16 |
| | Integer | INT, INTEGER, SERIAL | Int32 | IfxInt32 |
| | BigInt, BigSerial | BIGINT, BIGSERIAL, INT8, SERIAL8 | Int64 | IfxInt64 |
| | Real | REAL, SMALLFLOAT | Single | IfxReal |
| | Double | DECIMAL (≤ 29), DOUBLE PRECISION | Double | IfxDouble |
| | Float | DECIMAL (32), FLOAT | Double | IfxDouble |
| | Decimal | DECIMAL | Decimal | IfxDecimal |
| | Money | MONEY | Decimal | IfxDecimal |
| | Numeric | DECIMAL (≤ 29), NUMERIC | Decimal | IfxDecimal |

## Executing SQL statements from an application using the IBM Data Server Provider for .NET

When using the IBM Data Server Provider for .NET, the execution of SQL statements is done through a `IfxCommand` class using its methods `ExecuteReader()` and `ExecuteNonQuery()`, and its properties `CommandText`, `CommandType` and `Transaction`.

For SQL statements that produce output, the `ExecuteReader()` method should be used and its results can be retrieved from a `IfxDataReader` object. For all other SQL statements, the method `ExecuteNonQuery()` should be used. The `Transaction` property of the `IfxCommand` object should be initialized to a `IfxTransaction` object. A `IfxTransaction` object is responsible for rolling back and committing database transactions.

Executing an UPDATE statement in C#:

```
// assume a IfxConnection conn
IfxCommand cmd = conn.CreateCommand();
IfxTransaction trans = conn.BeginTransaction();
cmd.Transaction = trans;

cmd.CommandText = "UPDATE items " +
                  "  SET quantity = (SELECT MIN(quantity) " +
                  "                     FROM items " +
                  "                     WHERE item_num >= 5) " +
                  " WHERE id = 5"
cmd.ExecuteNonQuery();
```

Executing an UPDATE statement in Visual Basic .NET:

```
' assume a IfxConnection conn
IfxCommand cmd = conn.CreateCommand();
IfxTransaction trans = conn.BeginTransaction();
cmd.Transaction = trans;
cmd.CommandText = "UPDATE items " +
                  "  SET quantity = (SELECT MIN(quantity) " +
                  "                     FROM items " +
                  "                     WHERE item_num >= 5) " +
                  " WHERE id = 5"
cmd.ExecuteNonQuery();
```

Executing a SELECT statement in C#:

```
// assume a IfxConnection conn
IfxCommand cmd = conn.CreateCommand();
IfxTransaction trans = conn.BeginTransaction();
cmd.Transaction = trans;

cmd.CommandText = "SELECT item_num, manu_code " +
                  "  FROM itesm " +
                  "  WHERE item_num < 5";
DB2DataReader reader = cmd.ExecuteReader();
```

Executing a SELECT statement in Visual Basic .NET:

```
' assume a IfxConnection conn
Dim cmd As IfxCommand = conn.CreateCommand()
Dim trans As IfxTransaction = conn.BeginTransaction()
cmd.Transaction = trans

cmd.CommandText = "UPDATE items " +
                  "  SET quantity = (SELECT MIN(quantity) " +
                  "                     FROM items " +
                  "                     WHERE item_num >= 5) " +
                  " WHERE id = 5"
cmd.ExecuteNonQuery()
```

After your application has performed a database transaction, you must either roll it back or commit it. This is done through the `Commit()` and `Rollback()` methods of a `IfxTransaction` object.

Rolling back or committing a transaction in C#:

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

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

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

## 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 `IfxDataReader` object. The `IfxDataReader` method, `Read()` is used to advance to the next row in the result set.

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 IfxDataReader's Close() method is used to close the IfxDataReader object, which should always be done when reading the output is finished.

Reading a result set in C#:

```
// assume a IfxDataReader 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 IfxDataReader 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 an application using the IBM Data Server Provider for .NET

When using the IBM Data Server Provider for .NET, you can call stored procedures by using a IfxCommand object.

The default value of the CommandType property is CommandType.Text. This is the appropriate value for SQL statements and can also be used to call stored procedures. However, calling stored procedures is easier when you set CommandType to CommandType.StoredProcedure. In this case, you only need to specify the stored procedure name and any parameters.

When working with stored procedures you can pass in the parameters using host variables, named parameters, or positioned parameters. However, you cannot use them in combination within the same SQL statement.

The following C# and Visual Basic examples demonstrate how to invoke a stored procedure called INOUT_PARAM, with the CommandType property set to either CommandType.StoredProcedure or CommandType.Text.

- In C#, call a stored procedure by setting the CommandType property of the IfxCommand to CommandType.Text:

  ```
  // assume a IfxConnection conn
  IfxTransaction trans = conn.BeginTransaction();
  IfxCommand cmd = conn.CreateCommand();
  String procName = "INOUT_PARAM";
  String procCall = "EXECUTE PROCEDURE INOUT_PARAM (@param1, @param2, @param3)";
  cmd.Transaction = trans;
  cmd.CommandType = CommandType.Text;
  cmd.CommandText = procCall;
  ```

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

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

- In Visual Basic, call a stored procedure by setting the CommandType property of the IfxCommand to CommandType.Text:

```
' assume a IfxConnection conn
Dim trans As IfxTransaction = conn.BeginTransaction()
Dim cmd As IfxCommand = conn.CreateCommand()
Dim procName As String = "INOUT_PARAM"
Dim procCall As String = "EXECUTE PROCEDURE INOUT_PARAM (?, ?, ?)"
cmd.Transaction = trans
cmd.CommandType = CommandType.Text
cmd.CommandText = procCall

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

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

**Note:** Both CALL and EXECUTE PROCEDURE are supported.

- In C#, call a stored procedure by setting the CommandType property of the IfxCommand to CommandType.StoredProcedure. Named parameters are not supported when using this method:

```
// assume a IfxConnection conn
IfxTransaction trans = conn.BeginTransaction();
IfxCommand 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 IfxCommand
...

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

- In Visual Basic, call a stored procedure by setting the CommandType property of the IfxCommand to CommandType.StoredProcedure:

```
' assume a IfxConnection conn
Dim trans As IfxTransaction = conn.BeginTransaction()
Dim cmd As IfxCommand = 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 IfxCommand
...

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

- Starting in IBM DB2 Version 9.7 FixPack 3, a new member by name Cursor has been introduced. This member should be used when binding an output parameter of the type cursor: The example shows retrieving multiple data readers from the Parameter Collection

**.NET code 1**

```
conn.Open()

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();

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

**DDL**

```
CREATE OR REPLACE TYPE cur AS CURSOR
CREATE OR REPLACE PROCEDURE cursor_test (OUT curparam1 cur, OUT curparam2 cur)
  LANGUAGE SQL
  BEGIN
    SET curparam1 = CURSOR FOR SELECT * FROM curtest1;
    SET curparam2 = CURSOR FOR SELECT * FROM curtest2;
END
```

# Chapter 4. Building .NET Applications

## C# .NET application compile and link options

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

The following are the compile and link options recommended 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 bldapp |
| --- |
| **Compile and link options for standalone C# applications:**<br>**%BLDCOMP%**<br>Variable for the compiler. The default is `csc`, the Microsoft C# compiler.<br>**/r:<install_dir>\bin\%VERSION%IBM.Data.Informix.dll**<br>Reference the Informix dynamic link library for the .NET framework version you are using.<br>**%VERSION%**<br>There are several supported versions of the .NET framework for applications. Informix 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:**

**%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:<install_dir>\bin\%VERSION%IBM.Data.DB2.dll**
Reference the Informix dynamic link library for the .NET framework version you are using.

**%VERSION%**
There are several supported versions of the .NET framework for applications. Informix 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.

# Chapter 5. IBM Data Server Provider for .NET Reference

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

The IBM Data Server Provider for .NET allows your .NET applications to access the following database management systems:

- DB2 Universal Database Version 5 Release 4, Version 6 Release 1 and Version 7 Release 1 for iSeries, through DB2 Connect (for IBM DB2 Version 9.7 Fix Pack 4 and higher versions)
- DB2 Universal Database Version 5 Release 4 and Version 6 Release 1 for iSeries, through DB2 Connect (for IBM DB2 Version 9.7 Fix Pack 3 and earlier versions)
- IBM Informix Version 11.10, Version 11.50, and Version 11.70

In addition to the IBM Data Server Provider for .NET, the IBM Database Add-Ins for Visual Studio enables you to quickly and easily develop .NET applications for databases in Visual Studio. You can also use the Add-Ins to create database objects such as indexes and tables, and develop server-side objects, such as stored procedures and user-defined functions.

### System Requirements for the IBM Data Server Provider for .NET

Before using the Informix Install program to install the IBM Data Server Provider for .NET, you must already have the .NET Framework installed on your computer. If the .NET Framework is not installed, the Informix Install program will not install the IBM Data Server Provider for .NET.

### Using the IBM Data Server Provider for .NET Documentation

The IBM Data Server Provider for .NET reference documentation contains detailed provider-specific information. Many of the IBM Data Server Provider for .NET classes inherit or implement members from other .NET Framework classes or interfaces. The provider documentation includes a summary of all members within each of these classes. For more detailed information about a specific inherited member, see the appropriate topic in the Microsoft .NET Framework SDK.

### Namespaces

| Namespace | Description |
|---|---|
| IBM.Data.Informix | The .NET Framework data provider for Informix data servers. |

## IBM.Data.Informix Namespace

The IBM.Data.Informix namespace is the .NET Framework Data Provider for IBM data servers. The IBM Data Server Provider for .NET extends support for the ADO.NET interface and delivers high-performing, secure access to data. This provider is sometimes called the Common IDS .NET Provider.

The IBM Data Server Provider for .NET allows your .NET applications to access the following database management systems:

- DB2 Universal Database Version 5 Release 4, Version 6 Release 1 and Version 7 Release 1 for iSeries, through DB2 Connect (for IBM DB2 Version 9.7 Fix Pack 4 and higher versions)
- DB2 Universal Database Version 5 Release 4 and Version 6 Release 1 for iSeries, through DB2 Connect (for IBM DB2 Version 9.7 Fix Pack 3 and earlier versions)
- IBM Informix, Version 11.10 or later

In addition to the IBM Data Server Provider for .NET, the IBM Database Add-Ins for Visual Studio enable you to quickly and easily develop .NET applications for IBM data servers in Visual Studio 2005 or later. You can also use the Add-Ins to create database objects such as indexes and tables, and develop server-side objects, such as stored procedures and user-defined functions.

## Using the IBM Data Server Provider for .NET Documentation

The IBM Data Server Provider for .NET reference documentation contains detailed provider-specific information. Many of the IBM Data Server Provider for .NET classes inherit or implement members from other .NET Framework classes or interfaces. The provider documentation includes a summary of all members within each of these classes. For more detailed information about a specific inherited member, see the appropriate topic in the Microsoft .NET Framework SDK.

## Developing IBM Data Server Provider for .NET Applications

The IBM Data Server Provider for .NET provides functionality for connecting to a database, executing commands, and retrieving results. Those results can be processed directly, or placed in an ADO.NET DataSet for further processing while in a disconnected state. While in the DataSet, data can be exposed to the user, combined with other data from multiple sources, or passed remotely between tiers. Any processing performed on the data while in the DataSet can then be reconciled to the database.

To use the IBM Data Server Provider for .NET, you must add an `imports` or `using` statement for the IBM.Data.Informix namespace to your application .DLL. For example:

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

You also must include a reference to `IBM.Data.Informix.dll` in your application's project. In Visual Studio you can do this either from the References section for your project in the Solution Explorer, or from the **Project** > **Add Reference** menu option. If you are compiling a program from the command line for Framework 2.0 or later, your **csc** or **vbc** commands should include the following option:

```
/r:<install dir>\bin\netf20\IBM.Data.Informix.dll
```

To establish a connection to one of the supported data servers, you will need to construct a IfxConnection object and provide it with a valid Informix .NET connection string. See the IfxConnection.ConnectionString property for information on the supported keywords.

For detailed information about how to best use the IBM.Data.Informix namespace, see the documentation on the following IBM Data Server Provider for .NET classes:

- IfxDataAdapter

- IfxCommand
- IfxConnection
- IfxDataReader

## Classes

| Class | Description |
|---|---|
| IfxBulkCopy | Facilitates the copying of rows from one data source to another. |
| IfxBulkCopyColumnMapping | Represents a column mapping from the data source table to the destination table. |
| IfxBulkCopyColumnMappingCollection | Represents a collection of column mappings from the data source table to the destination table. |
| IfxCommand | Represents an SQL statement or stored procedure to execute against a database. |
| IfxCommandBuilder | Automatically generates single-table commands used to reconcile changes made to a DataSet with the associated database. |
| IfxConnection | Represents an open connection to a database. |
| IfxConnectionStringBuilder | Facilitates generic and IBM Data Server Provider for .NET-specific approaches to generating valid connection strings. |
| IfxDataAdapter | Represents a set of data commands and a connection to a database that are used to fill the DataSet and update the database. |
| IfxDataReader | Provides a way of reading a forward-only stream of data rows from a database. |
| IfxDataSourceEnumerator | Provides a way to discover visible Informix family data sources. |
| IfxError | Collects information relevant to a warning or error returned by the database. |
| IfxErrorCollection | Collects all errors generated by the IfxDataAdapter. |
| IfxException | The exception that is generated when an error is returned by an Informix data server. |
| IfxFactory | Represents a set of methods for creating instances of the System.Data.Common data source classes for the data provider. |
| IfxInfoMessageEventArgs | Provides data for the InfoMessage event. |
| IfxParameter | Represents a parameter to a IfxCommand and optionally, its mapping to a DataColumn. |
| IfxParameterCollection | Represents a collection of parameters relevant to a IfxCommand as well as their respective mappings to columns in a DataSet. |
| IfxPermission | Enables the IBM Data Server Provider for .NET to ensure that a user has a security level adequate to access an Informix database. |
| IfxPermissionAttribute | Associates a security action with a custom security attribute. |
| IfxRowsCopiedEventArgs | Provides data for the IfxRowsCopied event. |

| Class | Description |
|---|---|
| IfxRowUpdatedEventArgs | Provides data for the RowUpdated event. |
| IfxRowUpdatingEventArgs | Provides data for the RowUpdating event. |
| IfxTransaction | Represents an SQL transaction to be made at a database. |

### Delegates

| Delegate | Description |
|---|---|
| IfxInfoMessageEventHandler | Represents the method that will handle the InfoMessage event of a IfxConnection object. |
| IfxRowsCopiedEventHandler | Represents the method that will handle the IfxRowsCopied event of a IfxBulkCopy object. |
| IfxRowUpdatedEventHandler | Represents the method that will handle the RowUpdated event of a IfxDataAdapter object. |
| IfxRowUpdatingEventHandler | Represents the method that will handle the RowUpdating event of a IfxDataAdapter object. |

### Enumerations

| Enumeration | Description |
|---|---|
| IfxBulkCopyOptions | Specify options to use with IfxBulkCopy. These options are bit flags, which enable you to combine them in bit masks. |
| IfxType | Specifies the data type of a field, property, or IfxParameter. |

**Reference**

Chapter 5, “IBM Data Server Provider for .NET Reference,” on page 5-1

## IfxBinary Structure

Represents the BINARY, VARBINARY, CHAR FOR BIT DATA, VARCHAR FOR BIT DATA, and LONG VARCHAR FOR BIT DATA Informix data types. Encapsulates the byte[ ] .NET data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Structure IfxBinary
[C#]
public struct IfxBinary
[C++]
public value class IfxBinary
```

## Example

[C#] The following example demonstrates how to retrieve a single FOR BIT DATA character string column value from a table.

```
[C#]
  public static string getParam(IfxConnection conn)
  {
    string mySelectQuery = "SELECT BINARYCOL FROM TESTTBL";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);
    IfxDataReader reader = myCommand.ExecuteReader();

    if (reader.Read())
    {
      IfxBinary selectValue = reader.GetIfxBinary(0);

      if (!selectValue.IsNull) { return selectValue.ToString(); }
    }

    return "NULL";
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBinary Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxBinary Members

Represents the BINARY, VARBINARY, CHAR FOR BIT DATA, VARCHAR FOR BIT DATA, and LONG VARCHAR FOR BIT DATA Informix data types. Encapsulates the byte[ ] .NET data type. The following tables list the members exposed by the IfxBinary structure.

*Table 5-1. Public Fields*

| Public Field | Description |
|---|---|
|  Null | Null value for IfxBinary. |

*Table 5-2. Public Constructors*

| Public Constructor | Description |
|---|---|
|  IfxBinary | Initializes a new IfxBinary structure, with the specified byte array. |

### Public Properties

*Table 5-3. Public Properties*

| Public Property | Description |
|---|---|
|  IsNull | Gets a value that indicates if the value stored in the IfxBinary object is null. |
|  Value | Gets the value stored in the IfxBinary structure. |

*Table 5-4. Public Methods*

| Public Method | Description |
|---|---|
|  ToString | Returns a string that represents the IfxBinary structure. |
|  op_explicit | Converts the supplied IfxBinary structure to a byte array. |
|  op_implicit | Converts the supplied byte array to IfxBinary |

**Reference**

"IfxBinary Structure" on page 5-4

"IBM.Data.Informix Namespace" on page 5-1

## IfxBinary Fields

The fields of the IfxBinary class are listed here.

*Table 5-5. Public Fields*

| Public Field | Description |
|---|---|
|  Null | Null value for IfxBinary. |

**Reference**

"IfxBinary Members" on page 5-5

"IfxBinary Structure" on page 5-4

"IBM.Data.Informix Namespace" on page 5-1

**IfxBinary.Null Field:**

Null value for IfxBinary.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxBinary
[C#]
public static readonly IfxBinary Null
[C++]
public:
```

```
static initonly IfxBinary Null
[JScript]
public static final var Null () : IfxBinary
```

**Remarks**

The value of this constant is NULL.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBinary Structure" on page 5-4

"IBM.Data.Informix Namespace" on page 5-1

## IfxBinary Constructor

Initializes a new IfxBinary object, with the specified byte array.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New(value as Byte())
[C#]
public IfxBinary(byte[] value);
[C++]
public: IfxBinary(unsigned char value __gc[] );
[JScript]
public function IfxBinary(value : byte[]);
```

### Parameters

**value** A byte array to populate the IfxBinary instance.

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBinary Structure" on page 5-4

"IBM.Data.Informix Namespace" on page 5-1

## IfxBinary Methods

The methods of the IfxBinary structure are listed here.

*Table 5-6. Public Methods*

| Public Method | Description |
|---|---|
|  ToString | Returns a string that represents the IfxBinary structure. |
|  op_explicit | Converts the supplied IfxBinary structure to a byte array. |

*Table 5-6. Public Methods (continued)*

| Public Method | Description |
|---|---|
|  op_implicit | Converts the supplied byte array to IfxBinary |

**Reference**

"IfxBinary Members" on page 5-5

"IfxBinary Structure" on page 5-4

"IBM.Data.Informix Namespace" on page 5-1

**IfxBinary.op_explicit Method:**

Converts the supplied IfxBinary structure to a byte array.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Narrowing Operator CType (source As IfxBinary) As Byte()
[C#]
public static explicit operator byte[] (IfxBinary source)
[C++]
public:
static explicit operator array<unsigned char>^ (
 IfxBinary source
)
```

**Parameters**

**source** A IfxBinary structure to be converted to a byte array.

**Return value**

A byte array value converted from a IfxBinary instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBinary Structure" on page 5-4

"IBM.Data.Informix Namespace" on page 5-1

**IfxBinary.op_implicit Method:**

Converts the supplied byte array to IfxBinary.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Widening Operator CType (source As Byte()) As IfxBinary
[C#]
public static implicit operator IfxBinary (byte[] source)
[C++]
public:
static implicit operator IfxBinary (unsigned char source __gc[] )
```

**Parameters**

**source** A byte array to be converted to IfxBinary.

**Return value**

A IfxBinary structure with the value of **source**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBinary Structure" on page 5-4

"IBM.Data.Informix Namespace" on page 5-1

**IfxBinary.ToString Method:**

Returns a string that represents the IfxBinary structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string composed of hexadecimal values for each byte of the value of the IfxBinary structure.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBinary Structure" on page 5-4

"IBM.Data.Informix Namespace" on page 5-1

## IfxBinary Properties

The properties of the IfxBinary structure are listed here.

### Public Properties

*Table 5-7. Public Properties*

| Public Property | Description |
|---|---|
|  IsNull | Gets a value that indicates if the value stored in the IfxBinary object is null. |
|  Value | Gets the value stored in the IfxBinary structure. |

**Reference**

"IfxBinary Members" on page 5-5

"IfxBinary Structure" on page 5-4

"IBM.Data.Informix Namespace" on page 5-1

**IfxBinary.IsNull Property:**

Gets a value that indicates if the value stored in the IfxBinary structure is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBinary Structure" on page 5-4

"IBM.Data.Informix Namespace" on page 5-1

**IfxBinary.Value Property:**

Gets the value stored in the IfxBinary object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Value As Byte()
[C#]
public byte[] Value {get;}
[C++]
public: __property unsigned char value __gc[] get_Value();
[JScript]
public function get Value() : byte[];
```

**Property value**

A byte array representing the IfxBinary instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBinary Structure" on page 5-4

"IBM.Data.Informix Namespace" on page 5-1

# IfxBlob Class

Represents the BLOB Informix data type. Encapsulates the `byte[]` .NET data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
Public Structure IfxBlob
[C#]
public struct IfxBlob
[C++]
public value class IfxBlob
```

## Example

[C#] The following example demonstrates how to retrieve a single BLOB column value from a table.

```
[C#]
  public static IfxBlob getParam(IfxConnection conn)
  {
      string mySelectQuery = "SELECT * FROM EMP_PHOTO";
      IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);

      IfxDataReader reader = myCommand.ExecuteReader();

      if (reader.Read())
      {
          IfxBlob selectValue = reader.GetIfxBlob(2);

          if (!selectValue.IsNull) { return selectValue; }
```

```
        }

        return null;
    }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBlob Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxBlob Members

Represents the BLOB Informix data type. Encapsulates the byte[] .NET data type. The following tables list the members exposed by the IfxBlob class.

*Table 5-8. Public Fields*

| Public Field | Description |
|---|---|
| Null | Null value for IfxBlob. |

*Table 5-9. Public Constructors*

| Public Constructor | Description |
|---|---|
| IfxBlob | Initializes a new IfxBlob object with the specified byte array. |

*Table 5-10. Public Properties*

| Public Property | Description |
|---|---|
| CacheData | Indicates if the the data stored in the IfxBlob instance's current cursor position is being cached. |
| IsNull | Gets a value that indicates if the value stored in the IfxBlob object is null. |
| Value | Gets the value stored in the IfxBlob object. |

*Table 5-11. Public Methods*

| Public Method | Description |
|---|---|
| GetBytes | Reads a stream of bytes from the specified column offset into the buffer as an array, starting at the given buffer offset. |
| ToString | Returns a string that represents the IfxBlob structure. |

**Reference**

"IfxBlob Class" on page 5-11
"IBM.Data.Informix Namespace" on page 5-1

## IfxBlob Fields

The fields of the IfxBlob structure are listed here.

*Table 5-12. Public Fields*

| Public Field | Description |
|---|---|
|  Null | Null value for IfxBlob. |

**Reference**

"IfxBlob Members" on page 5-12
"IfxBlob Class" on page 5-11
"IBM.Data.Informix Namespace" on page 5-1

**IfxBlob.Null Field:**

Null value for IfxBlob.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxBlob
[C#]
public static readonly IfxBlob Null
[C++]
public:
static initonly IfxBlob Null
[JScript]
public static final var Null () : IfxBlob
```

**Remarks**

The value of this constant is `NULL`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBlob Class" on page 5-11
"IBM.Data.Informix Namespace" on page 5-1

## IfxBlob Constructor

Initializes a new IfxBlob structure with the specified byte array.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New(value as Byte())
[C#]
public IfxBlob(byte[] value);
[C++]
public: IfxBlob(unsigned char value __gc[]);
[JScript]
public function IfxBlob(value : byte[]);
```

### Parameters

**value** A byte array to populate the IfxBlob instance.

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBlob Class" on page 5-11

"IBM.Data.Informix Namespace" on page 5-1

## IfxBlob Methods

The methods of the IfxBlob structure are listed here.

*Table 5-13. Public Methods*

| Public Method | Description |
|---|---|
|  GetBytes | Reads a stream of bytes from the specified column offset into the buffer as an array, starting at the given buffer offset. |
|  ToString | Returns a string that represents the IfxBlob structure. |

**Reference**

"IfxBlob Members" on page 5-12

"IfxBlob Class" on page 5-11

"IBM.Data.Informix Namespace" on page 5-1

**IfxBlob.GetBytes Method:**

Reads a stream of bytes from the specified column offset into the buffer as an array, starting at the given buffer offset.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetBytes( _
   ByVal dataIndex As Long, _
   ByVal buffer() As Byte, _
   ByVal bufferIndex As Integer, _
   ByVal length As Integer _
) As Long
[C#]
```

```
public long GetBytes(
   long dataIndex,
   byte[] buffer,
   int bufferIndex,
   int length
);
[C++]
public: __int64 GetBytes(
   __int64 dataIndex,
   unsigned char buffer __gc[],
   int bufferIndex,
   int length
);
[JScript]
public function GetBytes(
   dataIndex : long,
   buffer : Byte[],
   bufferIndex : int,
   length : int
) : long;
```

**Parameters**

**dataIndex**
The index within the row where the read operation is to begin.

**buffer** The buffer into which to copy data.

**bufferIndex**
The index where **buffer** is to begin the write operation.

**length** The number of characters to read.

**Return value**

The actual number of bytes read.

**Exceptions**

| Exception type | Condition |
|---|---|
| IfxException | Invalid conversion. |
| InvalidOperationException | There is no more data to return. |

**Remarks**

GetBytes returns the number of available characters in the field. In most cases this is the exact length of the field. However, the number returned may be less than the true length of the field if GetBytes has already been used to obtain characters from the field. This may be the case, for example, if the IfxBlob is reading a BLOB into a buffer. For more information, see the SequentialAccess setting of System.Data.CommandBehavior in the Microsoft(R) .NET Framework SDK documentation.

If you pass a buffer that is a null value. GetBytes returns the length of the field in characters.

If the CacheData property is `false`, applications can only retrieve the complete set of data from a IfxBlob instance once. Any additional attempts at retrieving data will result in an InvalidOperationException. The specific implications of this are:

- This method can only be used to retrieve the complete set of data from a IfxBlob instance once.
- If you use this method to retrieve data from a IfxBlob instance, you cannot use the Value property or the ToString method to do the same.
- If you have already used the Value property or the ToString method to retrieve data from a IfxBlob instance, you cannot use this method to do the same.

If the CacheData property is `true`, data can be read from the DB2Blob structure, using any of the access methods.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBlob Class" on page 5-11

"IfxBlob Members" on page 5-12

"IBM.Data.Informix Namespace" on page 5-1

**IfxBlob.ToString Method:**

Returns a string that represents the IfxBlob structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxBlob structure.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidOperationException | There is no more data to return. |

**Remarks**

If the CacheData property is `false`, applications can only retrieve the complete set of data from a IfxBlob instance once. Any additional attempts at retrieving data will result in an InvalidOperationException. The specific implications of this are:

- This method can only be used to retrieve data from a IfxBlob instance once.

- If you use this method to retrieve data from a IfxBlob instance, you cannot use the Value property or the GetBytes method to do the same.
- If you have already used the Value property or the GetBytes method to retrieve data from a IfxBlob instance, you cannot use this method to do the same.

If the CacheData property is `true`, data can be read from the DB2Blob structure, using any of the access methods.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBlob Class" on page 5-11

"IBM.Data.Informix Namespace" on page 5-1

## IfxBlob Properties

The properties of the IfxBlob structure are listed here.

*Table 5-14. Public Properties*

| Public Property | Description |
|---|---|
| CacheData | Indicates if the the data stored in the IfxBlob instance's current cursor position is being cached. |
| IsNull | Gets a value that indicates if the value stored in the IfxBlob object is null. |
| Value | Gets the value stored in the IfxBlob object. |

**Reference**

"IfxBlob Members" on page 5-12

"IfxBlob Class" on page 5-11

"IBM.Data.Informix Namespace" on page 5-1

**IfxBlob.CacheData Property:**

Indicates if the data stored in the IfxBlob instance's current cursor position is being cached.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property CacheData As Boolean
[C#]
public bool CacheData {get;}
[C++]
public: __property bool get_CacheData();
[JScript]
public function get CacheData() : Boolean;
```

**Property value**

`true` if the IfxBlob structure's instance is being cached; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBlob Class" on page 5-11

"IfxBlob Members" on page 5-12

"IBM.Data.Informix Namespace" on page 5-1

**IfxBlob.IsNull Property:**

Gets a value that indicates if the value stored in the IfxBlob object is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBlob Class" on page 5-11

"IBM.Data.Informix Namespace" on page 5-1

**IfxBlob.Value Property:**

Gets the value stored in the IfxBlob structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Value As Byte()
[C#]
```

```
public byte[] Value {get;}
[C++]
public: __property unsigned char __gc[] get_Value();
[JScript]
public function get Value() : byte[];
```

**Property value**

A byte array representing the IfxBlob instance.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidOperationException | There is no more data to return. |

**Remarks**

If the CacheData property is `false`, applications can only retrieve the complete set of data from a IfxBlob instance once. Any additional attempts at retrieving data will result in an InvalidOperationException. The specific implications of this are:

- This property can only be used to retrieve data from a IfxBlob instance once.
- If you use this property to retrieve data from a IfxBlob instance, you cannot use the ToString or GetBytes methods to do the same.
- If you have already used the ToString or GetBytes methods to retrieve data from a IfxBlob instance, you cannot use this property to do the same.

If the CacheData property is `true`, data can be read from the DB2Blob structure, using any of the access methods.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBlob Class" on page 5-11

"IBM.Data.Informix Namespace" on page 5-1

# IfxBulkCopy Class

Facilitates the copying of rows from a data source to a table in a database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
  System.MarshalByRefObject
    IBM.Data.Informix.IfxBulkCopy

## Syntax

```
[Visual Basic]
NotInheritable Public Class IfxBulkCopy
[C#]
```

```
public sealed class IfxBulkCopy
[C++]
public ref class IfxBulkCopy sealed
[JScript]
public final class IfxBulkCopy
```

## Remarks

The source of the data can be DataRow, DataTable, or DataReader objects, which means that data to be copied can come from any source that is readable by your application. A value is generated automatically for such source of data using the GENERATED ALWAYS AS statement.The destination is a table in a database.

To enable the copying of data from columns that have names different from the destination table, IfxBulkCopy uses IfxCopyColumnMapping and IfxCopyColumnMappingCollection objects.

## Example

[C#] The following example demonstrates a bulk copy of data from a DataTable source into the SALES table.

```
[C#]
public static void copyIntoSales(IfxConnection conn, DataTable source)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "SALES";

  try
  {
    salesCopy.WriteToServer(source);
    salesCopy.Close();
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Members"

"IfxBulkCopyOptions Enumeration" on page 5-74

"IfxBulkCopyColumnMapping Class" on page 5-39

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopy Members

Facilitates the copying of rows from one data source to another. The following tables list the members exposed by the IfxBulkCopy class.

## Public Constructors

| Name | Description |
|---|---|
| IfxBulkCopy | Overloaded. Initializes a new instance of the IfxBulkCopy class. |

## Public Properties

| Name | Description |
|---|---|
| BulkCopyTimeout | Number of seconds for the bulk copy operation to complete before it times out. |
| ColumnMappings | Mappings of columns from the data source table to columns in the destination table. |
| DestinationTableName | Name of the destination table on the server. |
| Errors | Errors in the bulk copy operation. |
| NotifyAfter | The number of rows to copy before generating a IfxRowsCopied event. |

## Public Methods

| Name | Description |
|---|---|
| Close | Closes the IfxBulkCopy instance. |
| Dispose | Closes the IfxBulkCopy instance. |
| WriteToServer | Overloaded. Copies all rows from a data source to the destination table specified by the DestinationTableName property. |

## Public Events

| Name | Description |
|---|---|
| IfxRowsCopied | Occurs every time the number of rows copied reaches the value set in the NotifyAfter property. |

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

# IfxBulkCopy Constructors

Initializes a new instance of the IfxBulkCopy class.

## Overload List

| Method | Description |
|---|---|
|  IfxBulkCopy(IBM.Data.DB2.IfxConnection) | Initializes a new instance of the IfxBulkCopy class with a IfxConnection object. |
|  IfxBulkCopy(IBM.Data.DB2.IfxConnection, IBM.Data.DB2.IfxBulkCopyOptions) | Initializes a new instance of the IfxBulkCopy class with a IfxConnection object and IfxBulkCopyOptions enumeration values. |
|  IfxBulkCopy(string) | Initializes a new instance of the IfxBulkCopy class with a database connection string. |
|  IfxBulkCopy(string, IBM.Data.DB2.IfxBulkCopyOptions) | Initializes a new instance of the IfxBulkCopy class with a database connection string and IfxBulkCopyOptions enumeration values. |

**Reference**

"IfxBulkCopy Members" on page 5-20

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopy.IfxBulkCopy(IBM.Data.DB2.IfxConnection) Constructor:**

Initializes a new instance of the IfxBulkCopy class with a IfxConnection object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New(connection As IfxConnection)
[C#]
public IfxBulkCopy(IfxConnection connection);
[C++]
public: IfxBulkCopy(IfxConnection* connection);
[JScript]
public function IfxBulkCopy(connection : IfxConnection);
```

**Parameters**

**connection**
A IfxConnection object that represents a connection to a database.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidOperationException | This IfxBulkCopy constructor requires access to an open database connection. |

**Remarks**

Data will be copied to a table on the data server identified in the connection parameter.

### Example

[C#] The following example demonstrates a bulk copy of data from a DataTable source into the SALES table.

```
[C#]
public static void copyIntoSales(IfxConnection conn, DataTable source)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "SALES";

  try
  {
    salesCopy.WriteToServer(source);
    salesCopy.Close();
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopy.IfxBulkCopy(IBM.Data.DB2.IfxConnection, IBM.Data.DB2.IfxBulkCopyOptions) Constructor:

Initializes a new instance of the IfxBulkCopy class with a IfxConnection object and IfxBulkCopyOptions enumeration values.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New( _
               connection As IfxConnection, _
               options As IfxBulkCopyOptions _
)
[C#]
public IfxBulkCopy(
                  IfxConnection connection,
                  IfxBulkCopyOptions options,
);
[C++]
public: IfxBulkCopy(
                  IfxConnection* connection,
                  IfxBulkCopyOptions options,
);
[JScript]
public function IfxBulkCopy(
                           connection : IfxConnection,
                           options : IfxBulkCopyOptions
);
```

**Parameters**

**connection**
A IfxConnection object that represents a connection to a database.

**options**
Options to use during the bulk copy operation. This can be specified as an individual IfxBulkCopyOptions enumeration or a bitwise mask value indicating the desired options from the IfxBulkCopyOptions enumeration.

**Remarks**

Data will be copied to a table on the data server identified in the connection parameter.

**Example**

[C#] The following example demonstrates a bulk copy of data from a DataTable source into the SALES table, using the **TableLock** and **Truncate** values from the IfxBulkCopyOptions enumeration. All values in the target table will be cleared before the bulk copy operation begins, and the table will be locked during the bulk copy operation.

```
[C#]
public static void copyIntoSales(IfxConnection conn, DataTable source)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(connString,
         (IfxBulkCopyOptions)IfxBulkCopyOptions.TableLock |
         IfxBulkCopyOptions.Truncate);

  salesCopy.DestinationTableName = "SALES";

  try
  {
    salesCopy.WriteToServer(source);
    salesCopy.Close();
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopy.IfxBulkCopy(string) Constructor:**

Initializes a new instance of the IfxBulkCopy class with a database connection string.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New(connectionString As String)
[C#]
public IfxBulkCopy(string connectionString);
[C++]
public: IfxBulkCopy(String* connectionString);
[JScript]
public function IfxBulkCopy(connectionString : String);
```

### Parameters

**connectionString**
The connection string for the database connection.

### Remarks

A IfxConnection object will be initialized using the connection string information. The IfxConnection will be opened and data will be copied to the table. The IfxConnection will be automatically closed after the bulk copy operation.

### Example

[C#] The following example demonstrates a bulk copy of data from a DataTable source into the SALES table.

```
[C#]
public static void copyIntoSales(String connString, DataTable source)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(connString);
  salesCopy.DestinationTableName = "SALES";

  try
  {
    salesCopy.WriteToServer(source);
    salesCopy.Close();
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopy.IfxBulkCopy(string, IBM.Data.DB2.IfxBulkCopyOptions) Constructor:

Initializes a new instance of the IfxBulkCopy class with a database connection string and IfxBulkCopyOptions enumeration values.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New( _
               connectionString As String, _
               options As IfxBulkCopyOptions _
)
[C#]
public IfxBulkCopy(
                  string connectionString,
                  IfxBulkCopyOptions options,
);
[C++]
public: IfxBulkCopy(
                  String* connectionString,
                  IfxBulkCopyOptions options,
);
[JScript]
public function IfxBulkCopy(
                           connectionString : String,
                           options : IfxBulkCopyOptions
);
```

### Parameters

**connectionString**
The connection string for the database connection.

**options**
Options to use during the bulk copy operation. This can be specified as an individual IfxBulkCopyOptions enumeration or a bitwise mask value indicating the desired options from the IfxBulkCopyOptions enumeration.

### Remarks

A IfxConnection object will be initialized using the connection string information. The IfxConnection will be opened and data will be copied to the table. The IfxConnection will be automatically closed after the bulk copy operation.

### Example

[C#] The following example demonstrates a bulk copy of data from a DataTable source into the SALES table, using the **TableLock** and **Truncate** values from the IfxBulkCopyOptions enumeration. All values in the target table will be cleared before the bulk copy operation begins, and the table will be locked during the bulk copy operation.

```
[C#]
public static void copyIntoSales(IfxConnection connString, DataTable source)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(connString,
         (IfxBulkCopyOptions)IfxBulkCopyOptions.TableLock |
         IfxBulkCopyOptions.Truncate);
  salesCopy.DestinationTableName = "SALES";

  try
  {
    salesCopy.WriteToServer(source);
    salesCopy.Close();
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopy Methods

The methods of the IfxBulkCopy class are listed here.

### Public Methods

| Name | Description |
|---|---|
|  Close | Closes the IfxBulkCopy instance. |
|  Dispose | Closes the IfxBulkCopy instance. |
|  WriteToServer | Overloaded. Copies all rows from a data source to the destination table specified by the DestinationTableName property. |

**Reference**

"IfxBulkCopy Members" on page 5-20

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopy.Close Method:**

Closes the IfxBulkCopy instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub Close
[C#]
public void Close ()
[C++]
public:
void Close ()
[JScript]
public function Close ()
```

**Remarks**

After you call Close on a IfxBulkCopy instance, no other operation will succeed. Any actions involving members of a IfxBulkCopy instance after running Close will result in the data provider throwing an InvalidOperationException.

### Example

[C#] The following example demonstrates a bulk copy of data from a DataTable source into the SALES table.

```
[C#]
public static void copyIntoSales(IfxConnection conn, DataTable source)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "SALES";

  try
  {
    salesCopy.WriteToServer(source);
    salesCopy.Close();
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopy.Dispose Method:

Closes the IfxBulkCopy instance.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub Dispose
[C#]
public void Dispose ()
[C++]
public:
void Dispose ()
[JScript]
public function Dispose ()
```

### Remarks

After you call Dispose on a IfxBulkCopy instance, no other operation will succeed. Any actions with members of a IfxBulkCopy instance after a Dispose will result in the application throwing an InvalidOperationException.

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19
"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopy.WriteToServer Method:**

Copies all rows from a data source to the destination table specified by the DestinationTableName property.

**Overload List**

| Name | Description |
|---|---|
| WriteToServer(System.Data.DataRow[ ]) | Copies all rows from the DataRow array to the destination table specified by the DestinationTableName property. |
| WriteToServer(System.Data.DataTable) | Copies all rows from the DataTable to the destination table specified by the DestinationTableName property. |
| WriteToServer(System.Data.DataTable, System.Data.DataRowState) | Copies only the rows in the specified state (indicated in DataRowState) from the DataTable to the destination table specified by the DestinationTableName property. |
| WriteToServer(System.Data.IDataReader) | Copies all rows from the DataReader to the destination table specified by the DestinationTableName property. |

**Reference**

"IfxBulkCopy Class" on page 5-19
"IBM.Data.Informix Namespace" on page 5-1

*IfxBulkCopy.WriteToServer(System.Data.DataRow[ ]) Method:*

Copies all rows from the DataRow array to a destination table specified by the DestinationTableName property.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub WriteToServer ( dataRows As DataRow() )
[C#]
public void WriteToServer ( DataRow[] dataRows )
[C++]
public:
void WriteToServer ( array<DataRow*>* dataRows )
[JScript]
public function WriteToServer ( dataRows : DataRow[] )
```

**Parameters**

**dataRows**
An array of DataRow objects to copy to the destination table.

**Example**

[C#] The following example demonstrates a bulk copy of data from an array of DataRow objects into the SALES table.

```
[C#]
public static void copyIntoSales(IfxConnection conn, DataRow [] source)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "SALES";

  try
  {
    salesCopy.WriteToServer(source);
    salesCopy.Close();
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

*IfxBulkCopy.WriteToServer(System.Data.DataTable) Method:*

Copies all rows from the DataTable to the destination table specified by the DestinationTableName property.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub WriteToServer ( dataTable As DataTable )
[C#]
public void WriteToServer ( DataTable dataTable )
[C++]
public:
void WriteToServer ( DataTable* dataTable )
[JScript]
public function WriteToServer ( dataTable : DataTable )
```

**Parameters**

**dataTable**

A DataTable object whose rows will be copied to the destination table.

**Example**

[C#] The following example demonstrates a bulk copy of data from a DataTable source into the SALES table.

```
[C#]
public static void copyIntoSales(IfxConnection conn, DataTable source)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "SALES";
```

```
  try
  {
    salesCopy.WriteToServer(source);
    salesCopy.Close();
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

*IfxBulkCopy.WriteToServer(System.Data.DataTable, System.Data.DataRowState) Method:*

Copies only the rows in the specified state (indicated in DataRowState) from the DataTable to the destination table specified by the DestinationTableName property.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub WriteToServer ( _
                          dataTable As DataTable, _
                          rowState As DataRowState _
)
[C#]
public void WriteToServer (
                           DataTable dataTable,
                           DataRowState rowState
)
[C++]
public:
void WriteToServer (
                    DataTable* dataTable,
                    DataRowState rowState
)
[JScript]
public function WriteToServer (
                               dataTable : DataTable,
                               rowState : DataRowState
)
```

**Parameters**

**dataTable**
A DataTable object whose rows will be copied to the destination table.

**rowState**
The DataRowState enumeration value, specifying the state of rows to be copied to the destination table.

### Remarks

The allowable states that can be specified by DataRowState are **Added**, **Changed** and **Unmodified**.

### Example

[C#] The following example demonstrates a bulk copy of data from a DataTable source into the SALES table.

```
[C#]
public static void copyIntoSales(IfxConnection conn, DataTable source)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "SALES";

  try
  {
    salesCopy.WriteToServer(source, DataRowState.Unchanged);
    salesCopy.Close();
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

*IfxBulkCopy.WriteToServer(System.Data.IDataReader) Method:*

Copies all rows from the DataReader to a destination table specified by the DestinationTableName property.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub WriteToServer ( dataReader As IDataReader )
[C#]
public void WriteToServer ( IDataReader dataReader )
[C++]
public:
void WriteToServer ( IDataReader* dataReader )
[JScript]
public function WriteToServer ( dataReader : IDataReader )
```

### Parameters

**dataReader**

An IDataReader object whose rows will be copied to the destination table.

**Remarks**

To avoid an InvalidOperationException, the IfxBulkCopy instance needs a IfxConnection instance that is distinct from the one being used by the IfxDataReader.

**Example**

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the SALES table.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "SALES";

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopy Properties

The properties of the IfxBulkCopy class are listed here.

### Public Properties

| Name | Description |
|---|---|
| BulkCopyTimeout | Number of seconds for the bulk copy operation to complete before it times out. |
| ColumnMappings | Mappings of columns from the data source table to columns in the destination table. |
| DestinationTableName | Name of the destination table on the server. |
| Errors | Errors in the bulk copy operation. |
| NotifyAfter | The number of rows to copy before generating a IfxRowsCopied event. |

**Reference**

"IfxBulkCopy Members" on page 5-20

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopy.BulkCopyTimeout Property:**

Number of seconds for the bulk copy operation to complete before it times out.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property IfxBulkCopyTimeout As Integer
[C#]
public int IfxBulkCopyTimeout {get; set;}
[C++]
public:
property int BulkCopyTimeout {
         int get();
         void set (int value);
}
[JScript]
public function get IfxBulkCopyTimeout () : int;
public function set IfxBulkCopyTimeout (value : int);
```

**Property value**

The integer value of the IfxBulkCopyTimeout property. The default is zero, which means no timeout.

**Remarks**

If a IfxBulkCopyTimeout is provided, it will override the query timeout. Otherwise, the query timeout will take effect.

**Example**

[C#] The following example demonstrates a bulk copy of data from a DataTable source into the SALES table, using a timeout of 30 seconds.

```
[C#]
public static void copyIntoSales(IfxConnection conn, DataTable source)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "SALES";
  salesCopy.BulkCopyTimeout = 30;

  try
  {
    salesCopy.WriteToServer(source);
    salesCopy.Close();
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopy.ColumnMappings Property:**

Mappings of columns from the data source table to columns in the destination table.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ColumnMappings As IfxBulkCopyColumnMappingCollection
[C#]
public IfxBulkCopyColumnMappingCollection ColumnMappings {get; set;}
[C++]
public: __property IfxBulkCopyColumnMappingCollection* get_ColumnMappings();
public: __property void set_ColumnMappings(IfxBulkCopyColumnMappingCollection*);
[JScript]
public function get ColumnMappings() : IfxBulkCopyColumnMappingCollection;
public function set ColumnMappings(IfxBulkCopyColumnMappingCollection);
```

**Property value**

A collection of column mappings. By default, this is an empty collection.

**Remarks**

The mappings defined in the IfxBulkCopyColumnMappingCollection instance are validated in the `WriteToServer()` call.

**Example**

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined by a IfxBulkCopyColumnMappingCollection instance.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMappingCollection colMapCollection;
  colMapCollection = new IfxBulkCopyColumnMappingCollection();

  salesCopy.ColumnMappings = colMapCollection;

  colMapCollection.Add("DEPTNUMB", "DEPTNO");
  colMapCollection.Add("DEPTNAME", "DEPTNAME");
  colMapCollection.Add("MANAGER", "ADMRDEPT");
```

```
  colMapCollection.Add("LOCATION", "LOCATION");

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopy.DestinationTableName Property:**

Name of the destination table on the server, where rows from the data source will be copied.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property DestinationTableName As String
[C#]
public string DestinationTableName {get; set;}
[C++]
public: __property String* get_DestinationTableName();
public: __property void set_DestinationTableName(String*);
[JScript]
public function get DestinationTableName() : String;
public function set DestinationTableName(String);
```

**Property value**

The string value of the destination table's name.

**Remarks**

A DestinationTableName must be set for the WriteToServer operation to be successful. If case sensitivity needs to be maintained for the destination table name, you need to enclose the destination table name in an extra set of double quotes. For example:

```
salesCopy.DestinationTable = "\"CanadaSales\"";
```

The value you provide for DestinationTableName can optionally include the database name and the schema name, in addition to the mandatory table name. For example: `SAMPLE.USER.SALES`.

### Example

[C#] The following example demonstrates a bulk copy of data from a DataTable source into the SALES table.

```
[C#]
public static void copyIntoSales(IfxConnection conn, DataTable source)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "SALES";

  try
  {
    salesCopy.WriteToServer(source);
    salesCopy.Close();
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopy.Errors Property:

Errors in the bulk copy operation.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Property Errors As IfxErrorCollection
[C#]
public IfxErrorCollection Errors {get;}
[C++]
public: __property IfxErrorCollection* get_Errors();
[JScript]
public function get Errors() : IfxErrorCollection;
```

### Property value

A IfxErrorCollection instance, representing a collection of errors that occurred during the bulk copy operation.

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopy.NotifyAfter Property:**

The number of rows to copy before generating a IfxRowsCopied event.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property NotifyAfter As Integer
[C#]
public int NotifyAfter {get; set;}
[C++]
public: __property int get_NotifyAfter();
public: __property void set_NotifyAfter(int);
[JScript]
public function get NotifyAfter() : int;
public function set NotifyAfter(int);
```

**Property value**

The integer value of the NotifyAfter property, which indicates the number of rows to copy to the destination table before generating a IfxRowsCopied event. The NotifyAfter property can be set from the IfxRowsCopied event handler.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19
"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopy Events

The events of the IfxBulkCopy class are listed here.

### Public Events

| Name | Description |
|---|---|
| IfxRowsCopied | Occurs every time the number of rows copied reaches the value set in the NotifyAfter property. |

**Reference**

"IfxBulkCopy Members" on page 5-20
"IfxBulkCopy Class" on page 5-19
"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopy.IfxRowsCopied Event:**

Occurs every time the number of rows copied reaches the value set in the NotifyAfter property.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Event IfxRowsCopied As IfxRowsCopiedEventHandler
[C#]
public event IfxRowsCopiedEventHandler IfxRowsCopied;
[C++]
public:
event IfxRowsCopiedEventHandler* IfxRowsCopied{
        void add (IfxRowsCopiedEventHandler* value);
        void remove (IfxRowsCopiedEventHandler* value);
}
```

[JScript] In JScript(R), you can handle the events defined by a class, but you cannot declare new events.

**Remarks**

The application can abort the operation in the event handler by setting the Abort property of IfxRowsCopiedEventArgs to `true`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IfxBulkCopy.NotifyAfter Property" on page 5-38

"IBM.Data.Informix Namespace" on page 5-1

# IfxBulkCopyColumnMapping Class

Represents a column mapping from the data source table to the destination table.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
  System.MarshalByRefObject
    IBM.Data.Informix.IfxBulkCopyColumnMapping

## Syntax

```
[Visual Basic]
NotInheritable Public Class IfxBulkCopyColumnMapping
[C#]
public sealed class IfxBulkCopyColumnMapping
[C++]
public __gc __sealed class IfxBulkCopyColumnMapping
[JScript]
public final class IfxBulkCopyColumnMapping
```

## Remarks

If no column mappings are defined, columns are mapped based on their ordinal positions in their respective table schema. If the source and target columns are not compatible, an InvalidOperationException is thrown.

## Example

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined by IfxBulkCopyColumnMapping instances.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMapping colMapDeptNum =
  new IfxBulkCopyColumnMapping("DEPTNUMB", "DEPTNO");
  IfxBulkCopyColumnMapping colMapDeptNme =
  new IfxBulkCopyColumnMapping("DEPTNAME", "DEPTNAME");
  IfxBulkCopyColumnMapping colMapDeptMgr =
  new IfxBulkCopyColumnMapping("MANAGER", "ADMRDEPT");
  IfxBulkCopyColumnMapping colMapDeptLoc =
  new IfxBulkCopyColumnMapping("LOCATION", "LOCATION");

  salesCopy.ColumnMappings.Add(colMapDeptNum);
  salesCopy.ColumnMappings.Add(colMapDeptNme);
  salesCopy.ColumnMappings.Add(colMapDeptMgr);
  salesCopy.ColumnMappings.Add(colMapDeptLoc);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopyColumnMapping Members" on page 5-41

"IfxBulkCopy Class" on page 5-19

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopyColumnMapping Members

Represents a column mapping from the data source table to the destination table. The following tables list the members exposed by the IfxBulkCopyColumnMapping class.

### Public Constructors

| Name | Description |
|---|---|
| IfxBulkCopyColumnMapping | Overloaded. Initializes a new IfxBulkCopyColumnMapping object. |

### Public Properties

| Name | Description |
|---|---|
| DestinationColumn | Name of the column being mapped in the destination table. |
| DestinationOrdinal | Ordinal value of the column being mapped in the destination table. |
| SourceColumn | Name of the column being copied from the data source. |
| SourceOrdinal | Ordinal value of the column being copied from the data source. |

### Public Methods

| Name | Description |
|---|---|
| Equals (Inherited from Object.) | Determines whether two IfxBulkCopyColumnMapping instances are equal. |
| GetHashCode (Inherited from Object.) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| ToString (Inherited from Object.) | Gets the complete text of the column mapping. |

**Reference**

"IfxBulkCopyColumnMapping Class" on page 5-39

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopyColumnMapping Constructors

Initializes a new instance of the IfxBulkCopyColumnMapping class.

### Overload List

| Name | Description |
|---|---|
| IfxBulkCopyColumnMapping() | Default constructor that initializes a new IfxBulkCopyColumnMapping object. |

| Name | Description |
|---|---|
|  IfxBulkCopyColumnMapping(int, int) | Initializes a new IfxBulkCopyColumnMapping object using column ordinals for the data source and destination columns. |
|  IfxBulkCopyColumnMapping(int, string) | Initializes a new IfxBulkCopyColumnMapping object using a column ordinal for the data source column and a column name for the destination column. |
|  IfxBulkCopyColumnMapping(string, int) | Initializes a new IfxBulkCopyColumnMapping object using a column name for the data source column and a column ordinal for the destination column. |
|  IfxBulkCopyColumnMapping(string, string) | Initializes a new IfxBulkCopyColumnMapping object using column names for the data source and destination columns. |

**Reference**

"IfxBulkCopyColumnMapping Members" on page 5-41
"IfxBulkCopyColumnMapping Class" on page 5-39
"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopyColumnMapping.IfxBulkCopyColumnMapping() Constructor:**

Default constructor that initializes a new IfxBulkCopyColumnMapping object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New
[C#]
public IfxBulkCopyColumnMapping();
[C++]
public: IfxBulkCopyColumnMapping();
[JScript]
public function IfxBulkCopyColumnMapping();
```

**Remarks**

If you use this constructor to create a IfxBulkCopyColumnMapping instance, you will need to assign source and destination columns. Source columns can be defined by the SourceColumn and SourceOrdinal properties. Destination columns can be defined by the DestinationColumn and DestinationOrdinal properties.

**Example**

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined in IfxBulkCopyColumnMapping instances by SourceColumn and DestinationColumn properties.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMapping colMapDeptNum = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptNme = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptMgr = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptLoc = new IfxBulkCopyColumnMapping();

  colMapDeptNum.DestinationColumn = "DEPTNO";
  colMapDeptNme.DestinationColumn = "DEPTNAME";
  colMapDeptMgr.DestinationColumn = "ADMRDEPT";
  colMapDeptLoc.DestinationColumn = "LOCATION";

  colMapDeptNum.SourceColumn = "DEPTNUMB";
  colMapDeptNme.SourceColumn = "DEPTNAME";
  colMapDeptMgr.SourceColumn = "MANAGER";
  colMapDeptLoc.SourceColumn = "LOCATION";

  salesCopy.ColumnMappings.Add(colMapDeptNum);
  salesCopy.ColumnMappings.Add(colMapDeptNme);
  salesCopy.ColumnMappings.Add(colMapDeptMgr);
  salesCopy.ColumnMappings.Add(colMapDeptLoc);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMapping Class" on page 5-39

"IBM.Data.Informix Namespace" on page 5-1

### IfxBulkCopyColumnMapping.IfxBulkCopyColumnMapping(int, int) Constructor:

Initializes a new IfxBulkCopyColumnMapping object using column ordinals for the data source and destination columns.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New ( _
  source As Integer, _
  destination As Integer _
)
```

```
[C#]
public IfxBulkCopyColumnMapping(
  int source
  int destination
);
[C++]
public: IfxBulkCopyColumnMapping(
  int source
  int destination
);
[JScript]
public function IfxBulkCopyColumnMapping(
  source : int,
  destination : int
)
```

### Parameters

**source** Ordinal value of the column being copied from the data source.

**destination**
Ordinal value of the column being mapped in the destination table.

### Example

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined in IfxBulkCopyColumnMapping instances.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMapping colMapDeptNum = new IfxBulkCopyColumnMapping(0,0);
  IfxBulkCopyColumnMapping colMapDeptNme = new IfxBulkCopyColumnMapping(1,1);
  IfxBulkCopyColumnMapping colMapDeptMgr = new IfxBulkCopyColumnMapping(2,3);
  IfxBulkCopyColumnMapping colMapDeptLoc = new IfxBulkCopyColumnMapping(4,4);

  salesCopy.ColumnMappings.Add(colMapDeptNum);
  salesCopy.ColumnMappings.Add(colMapDeptNme);
  salesCopy.ColumnMappings.Add(colMapDeptMgr);
  salesCopy.ColumnMappings.Add(colMapDeptLoc);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMapping Class" on page 5-39
"IBM.Data.Informix Namespace" on page 5-1

### IfxBulkCopyColumnMapping.IfxBulkCopyColumnMapping(int, string) Constructor:

Initializes a new IfxBulkCopyColumnMapping object using a column ordinal for the data source column and a column name for the destination column.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New ( _
  source As Integer, _
  destination As String _
)
[C#]
public IfxBulkCopyColumnMapping(
  int source
  string destination
);
[C++]
public: IfxBulkCopyColumnMapping(
  int source
  String* destination
);
[JScript]
public function IfxBulkCopyColumnMapping(
  source : int,
  destination : String
)
```

**Parameters**

**source** Ordinal value of the column being copied from the data source.

**destination**
Name of the column being mapped in the destination table.

**Remarks**

For case sensitivity to be maintained, the column name must be enclosed within an extra set of double-quotes. For example, the destination column can be set as follows:

```
IfxBulkCopyColumnMapping colMapDeptNum =
new IfxBulkCopyColumnMapping(0, "\"LoCaTion\"");
```

**Example**

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined in IfxBulkCopyColumnMapping instances.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMapping colMapDeptNum =
  new IfxBulkCopyColumnMapping(0, "\"DEPTNO\"");
```

```
  IfxBulkCopyColumnMapping colMapDeptNme =
  new IfxBulkCopyColumnMapping(1, "\"DEPTNAME\"");
  IfxBulkCopyColumnMapping colMapDeptMgr =
  new IfxBulkCopyColumnMapping(2, "\"ADMRDEPT\"");
  IfxBulkCopyColumnMapping colMapDeptLoc =
  new IfxBulkCopyColumnMapping(4, "\"LOCATION\"");

  salesCopy.ColumnMappings.Add(colMapDeptNum);
  salesCopy.ColumnMappings.Add(colMapDeptNme);
  salesCopy.ColumnMappings.Add(colMapDeptMgr);
  salesCopy.ColumnMappings.Add(colMapDeptLoc);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMapping Class" on page 5-39

"IBM.Data.Informix Namespace" on page 5-1

### IfxBulkCopyColumnMapping.IfxBulkCopyColumnMapping(string, int) Constructor:

Initializes a new IfxBulkCopyColumnMapping object using a column name for the data source column and a column ordinal for the destination column.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New ( _
  source As String, _
  destination As Integer _
)
[C#]
public IfxBulkCopyColumnMapping(
  string source
  int destination
);
[C++]
public: IfxBulkCopyColumnMapping(
  String* source
  int destination
);
[JScript]
```

```
public function IfxBulkCopyColumnMapping(
  source : String,
  destination : int
)
```

**Parameters**

**source** Name of the column being copied from the data source.

**destination**
Ordinal value of the column being mapped in the destination table.

**Remarks**

For case sensitivity to be maintained, the column name must be enclosed within an extra set of double-quotes. For example, the source column can be set as follows:

```
IfxBulkCopyColumnMapping colMapDeptNum =
new IfxBulkCopyColumnMapping("\"LoCaTion\"", 0);
```

**Example**

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined in IfxBulkCopyColumnMapping instances.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy =
  new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMapping colMapDeptNum =
  new IfxBulkCopyColumnMapping("\"DEPTNUMB\"", 0);
  IfxBulkCopyColumnMapping colMapDeptNme =
  new IfxBulkCopyColumnMapping("\"DEPTNAME\"", 1);
  IfxBulkCopyColumnMapping colMapDeptMgr =
  new IfxBulkCopyColumnMapping("\"MANAGER\"", 3);
  IfxBulkCopyColumnMapping colMapDeptLoc =
  new IfxBulkCopyColumnMapping("\"LOCATION\"", 4);

  salesCopy.ColumnMappings.Add(colMapDeptNum);
  salesCopy.ColumnMappings.Add(colMapDeptNme);
  salesCopy.ColumnMappings.Add(colMapDeptMgr);
  salesCopy.ColumnMappings.Add(colMapDeptLoc);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMapping Class" on page 5-39

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopyColumnMapping.IfxBulkCopyColumnMapping(string, string) Constructor:**

Initializes a new IfxBulkCopyColumnMapping object using column names for the data source and destination columns.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New ( _
  source As String, _
  destination As String _
)
[C#]
public IfxBulkCopyColumnMapping(
  string source
  string destination
);
[C++]
public: IfxBulkCopyColumnMapping(
  String* source
  String* destination
);
[JScript]
public function IfxBulkCopyColumnMapping(
  source : String,
  destination : String
)
```

**Parameters**

**source** Name of the column being copied from the data source.

**destination**
Name of the column being mapped in the destination table.

**Remarks**

For case sensitivity to be maintained, the column name must be enclosed within an extra set of double-quotes. For example, the source and destination columns can be set as follows:

```
IfxBulkCopyColumnMapping colMapDeptNum =
new IfxBulkCopyColumnMapping("\"LoCaTion1\"", "\"LoCaTion2\"");
```

**Example**

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined in IfxBulkCopyColumnMapping instances.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy =
  new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";
```

```
  IfxBulkCopyColumnMapping colMapDeptNum =
  new IfxBulkCopyColumnMapping("\"DEPTNUMB\"", "\"DEPTNO\"");
  IfxBulkCopyColumnMapping colMapDeptNme =
  new IfxBulkCopyColumnMapping("\"DEPTNAME\"", "\"DEPTNAME\"");
  IfxBulkCopyColumnMapping colMapDeptMgr =
  new IfxBulkCopyColumnMapping("\"MANAGER\"", "\"ADMRDEPT\"");
  IfxBulkCopyColumnMapping colMapDeptLoc =
  new IfxBulkCopyColumnMapping("\"LOCATION\"", "\"LOCATION\"");

  salesCopy.ColumnMappings.Add(colMapDeptNum);
  salesCopy.ColumnMappings.Add(colMapDeptNme);
  salesCopy.ColumnMappings.Add(colMapDeptMgr);
  salesCopy.ColumnMappings.Add(colMapDeptLoc);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMapping Class" on page 5-39

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopyColumnMapping Methods

The methods of the IfxBulkCopyColumnMapping class are listed here.

### Public Methods

| Name | Description |
|---|---|
| Equals (Inherited from Object.) | Determines whether two IfxBulkCopyColumnMapping instances are equal. |
| GetHashCode (Inherited from Object.) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| ToString (Inherited from Object.) | Gets the complete text of the column mapping. |

**Reference**

"IfxBulkCopyColumnMapping Members" on page 5-41

"IfxBulkCopyColumnMapping Class" on page 5-39

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopyColumnMapping Properties

The properties of the IfxBulkCopyColumnMapping class are listed here.

## Public Properties

| Name | Description |
|---|---|
| DestinationColumn | Name of the column being mapped in the destination table. |
| DestinationOrdinal | Ordinal value of the column being mapped in the destination table. |
| SourceColumn | Name of the column being copied from the data source. |
| SourceOrdinal | Ordinal value of the column being copied from the data source. |

**Reference**

"IfxBulkCopyColumnMapping Members" on page 5-41
"IfxBulkCopyColumnMapping Class" on page 5-39
"IBM.Data.Informix Namespace" on page 5-1

### IfxBulkCopyColumnMapping.DestinationColumn Property:

Name of the column being mapped in the destination table.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property DestinationColumn As String
[C#]
public string DestinationColumn {get; set;}
[C++]
public: __property String* get_DestinationColumn();
public: __property void set_DestinationColumn(String*);
[JScript]
public function get DestinationColumn() : String;
public function set DestinationColumn(String);
```

**Property value**

The string value of the destination column name.

**Remarks**

In the case where a destination column value is set multiple times (if values are added multiple times to either the DestinationColumn or DestinationOrdinal properties, or if a value is assigned at least once in each of these two properties), the last value set is what is used in the Add method.

For case sensitivity to be maintained, the column name must be enclosed within an extra set of double-quotes. For example, the destination column can be set as follows:

```
salesCopy.ColumnMappings.DestinationColumn= "\"LoCaTion\"";
```

### Example

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined in IfxBulkCopyColumnMapping instances by SourceColumn and DestinationColumn properties.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMapping colMapDeptNum = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptNme = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptMgr = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptLoc = new IfxBulkCopyColumnMapping();

  colMapDeptNum.DestinationColumn = "DEPTNO";
  colMapDeptNme.DestinationColumn = "DEPTNAME";
  colMapDeptMgr.DestinationColumn = "ADMRDEPT";
  colMapDeptLoc.DestinationColumn = "LOCATION";

  colMapDeptNum.SourceColumn = "DEPTNUMB";
  colMapDeptNme.SourceColumn = "DEPTNAME";
  colMapDeptMgr.SourceColumn = "MANAGER";
  colMapDeptLoc.SourceColumn = "LOCATION";

  salesCopy.ColumnMappings.Add(colMapDeptNum);
  salesCopy.ColumnMappings.Add(colMapDeptNme);
  salesCopy.ColumnMappings.Add(colMapDeptMgr);
  salesCopy.ColumnMappings.Add(colMapDeptLoc);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMapping Class" on page 5-39

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopyColumnMapping.DestinationOrdinal Property:

Ordinal value of the column being mapped in the destination table.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Property DestinationOrdinal As Integer
[C#]
public int DestinationOrdinal {get; set;}
[C++]
public: __property int get_DestinationOrdinal();
public: __property void set_DestinationOrdinal(int);
[JScript]
public function get DestinationOrdinal() : int;
public function set DestinationOrdinal(int);
```

### Property value

The integer value of the ordinal representing the destination column.

### Remarks

In the case where a destination column value is set multiple times (if values are added multiple times to either the DestinationColumn or DestinationOrdinal properties, or if a value is assigned at least once in each of these two properties), the last value set is what is used in the Add method.

### Example

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined in IfxBulkCopyColumnMapping instances by SourceOrdinal and DestinationOrdinal properties.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMapping colMapDeptNum = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptNme = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptMgr = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptLoc = new IfxBulkCopyColumnMapping();

  colMapDeptNum.DestinationOrdinal = 0;
  colMapDeptNme.DestinationOrdinal = 1;
  colMapDeptMgr.DestinationOrdinal = 3;
  colMapDeptLoc.DestinationOrdinal = 4;

  colMapDeptNum.SourceOrdinal = 0;
  colMapDeptNme.SourceOrdinal = 1;
  colMapDeptMgr.SourceOrdinal = 2;
  colMapDeptLoc.SourceOrdinal = 4;


  salesCopy.ColumnMappings.Add(colMapDeptNum);
  salesCopy.ColumnMappings.Add(colMapDeptNme);
  salesCopy.ColumnMappings.Add(colMapDeptMgr);
  salesCopy.ColumnMappings.Add(colMapDeptLoc);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
```

```
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMapping Class" on page 5-39

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopyColumnMapping.SourceColumn Property:**

Name of the column being copied from the data source.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property SourceColumn As String
[C#]
public string SourceColumn {get; set;}
[C++]
public: __property String* get_SourceColumn();
public: __property void set_SourceColumn(String*);
[JScript]
public function get SourceColumn() : String;
public function set SourceColumn(String);
```

**Property value**

The string value of the source column name.

**Remarks**

In the case where a source column value is set multiple times (if values are added multiple times to either the SourceColumn or SourceOrdinal properties, or if a value is assigned at least once in each of these two properties), the last value set is what is used in the Add method.

For case sensitivity to be maintained, the column name must be enclosed within an extra set of double-quotes. For example, the source column can be set as follows:

```
salesCopy.ColumnMappings.SourceColumn= "\"LoCaTion\"";
```

**Example**

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined in IfxBulkCopyColumnMapping instances by SourceColumn and DestinationColumn properties.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
```

```
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMapping colMapDeptNum = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptNme = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptMgr = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptLoc = new IfxBulkCopyColumnMapping();

  colMapDeptNum.DestinationColumn = "DEPTNO";
  colMapDeptNme.DestinationColumn = "DEPTNAME";
  colMapDeptMgr.DestinationColumn = "ADMRDEPT";
  colMapDeptLoc.DestinationColumn = "LOCATION";

  colMapDeptNum.SourceColumn = "DEPTNUMB";
  colMapDeptNme.SourceColumn = "DEPTNAME";
  colMapDeptMgr.SourceColumn = "MANAGER";
  colMapDeptLoc.SourceColumn = "LOCATION";

  salesCopy.ColumnMappings.Add(colMapDeptNum);
  salesCopy.ColumnMappings.Add(colMapDeptNme);
  salesCopy.ColumnMappings.Add(colMapDeptMgr);
  salesCopy.ColumnMappings.Add(colMapDeptLoc);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMapping Class" on page 5-39

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopyColumnMapping.SourceOrdinal Property:**

Ordinal value of the column being copied from the data source.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property SourceOrdinal As Integer
[C#]
public int SourceOrdinal {get; set;}
[C++]
public: __property int get_SourceOrdinal();
public: __property void set_SourceOrdinal(int);
[JScript]
public function get SourceOrdinal() : int;
public function set SourceOrdinal(int);
```

### Property value

The integer value of the ordinal representing the source column.

### Remarks

In the case where a source column value is set multiple times (if values are added multiple times to either the SourceColumn or SourceOrdinal properties, or if a value is assigned at least once in each of these two properties), the last value set is what is used in the Add method.

### Example

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined in IfxBulkCopyColumnMapping instances by SourceOrdinal and DestinationOrdinal properties.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMapping colMapDeptNum = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptNme = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptMgr = new IfxBulkCopyColumnMapping();
  IfxBulkCopyColumnMapping colMapDeptLoc = new IfxBulkCopyColumnMapping();

  colMapDeptNum.DestinationOrdinal = 0;
  colMapDeptNme.DestinationOrdinal = 1;
  colMapDeptMgr.DestinationOrdinal = 3;
  colMapDeptLoc.DestinationOrdinal = 4;

  colMapDeptNum.SourceOrdinal = 0;
  colMapDeptNme.SourceOrdinal = 1;
  colMapDeptMgr.SourceOrdinal = 2;
  colMapDeptLoc.SourceOrdinal = 4;


  salesCopy.ColumnMappings.Add(colMapDeptNum);
  salesCopy.ColumnMappings.Add(colMapDeptNme);
  salesCopy.ColumnMappings.Add(colMapDeptMgr);
  salesCopy.ColumnMappings.Add(colMapDeptLoc);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMapping Class" on page 5-39

"IBM.Data.Informix Namespace" on page 5-1

# IfxBulkCopyColumnMappingCollection Class

Represents a collection of column mappings from the data source table to the destination table.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
  System.Collections.CollectionBase
    IBM.Data.Informix.IfxBulkCopyColumnMappingCollection

## Syntax

```
[Visual Basic]
NotInheritable Public Class IfxBulkCopyColumnMappingCollection _
               Inherits System.Collections.CollectionBase
[C#]
public sealed class IfxBulkCopyColumnMappingCollection :
  System.Collections.CollectionBase
[C++]
public ref class IfxBulkCopyColumnMappingCollection sealed :
  public System.Collections.CollectionBase
[JScript]
public final class IfxBulkCopyColumnMappingCollection
  extends System.Collections.CollectionBase
```

## Remarks

If no column mappings are defined, columns are mapped based on their ordinal positions in their respective table schema. If the source and target columns are not compatible, an InvalidOperationException is thrown.

## Example

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined by a IfxBulkCopyColumnMappingCollection instance.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMappingCollection colMapCollection;
  colMapCollection = new IfxBulkCopyColumnMappingCollection();

  salesCopy.ColumnMappings = colMapCollection;

  colMapCollection.Add("DEPTNUMB", "DEPTNO");
  colMapCollection.Add("DEPTNAME", "DEPTNAME");
  colMapCollection.Add("MANAGER", "ADMRDEPT");
  colMapCollection.Add("LOCATION", "LOCATION");

  try
  {
    salesCopy.WriteToServer(reader);
```

```
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Members"

"IfxBulkCopy Class" on page 5-19

"IfxBulkCopyColumnMapping Class" on page 5-39

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopyColumnMappingCollection Members

Represents a collection of column mappings from the data source table to the destination table. The following tables list the members exposed by the IfxBulkCopyColumnMappingCollection class.

### Public Constructors

| Name | Description |
|---|---|
| IfxBulkCopyColumnMappingCollection | Initializes a new IfxBulkCopyColumnMappingCollection object. |

### Public Methods

| Name | Description |
|---|---|
| Add | Overloaded. Creates a IfxBulkCopyColumnMapping using column names for the data source and destination columns, and adds it to the IfxBulkCopyColumnMappingCollection. |
| Clear | Clears the current collection of IfxBulkCopyColumnMapping objects. |
| Contains | Returns a value indicating if the specified IfxBulkCopyColumnMapping object is in the IfxBulkCopyColumnMappingCollection. |
| CopyTo | Copies the IfxBulkCopyColumnMapping objects in the IfxBulkCopyColumnMappingCollection to an array of IfxBulkCopyColumnMapping objects, starting at a specified index of the array. |
| IndexOf | Returns a value indicating the position in the IfxBulkCopyColumnMappingCollection where the specified IfxBulkCopyColumnMapping object is located. |

| Name | Description |
|---|---|
| Insert | Inserts the specified IfxBulkCopyColumnMapping to the IfxBulkCopyColumnMappingCollection at the specified index position. |
| Remove | Removes the specified IfxBulkCopyColumnMapping from the IfxBulkCopyColumnMappingCollection. |
| RemoveAt | Removes the IfxBulkCopyColumnMapping at the specified index position from the IfxBulkCopyColumnMappingCollection. |
| ToString (Inherited from Object.) | Gets the complete text of the column mappings in the IfxBulkCopyColumnMappingCollection. |

### Protected Methods

| Name | Description |
|---|---|
| OnInsert (Inherited from CollectionBase.) | Performs additional custom processes before inserting a new mapping into the IfxBulkCopyColumnMappingCollection instance. |

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopyColumnMappingCollection Constructor

Initializes a new IfxBulkCopyColumnMappingCollection object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New
[C#]
public IfxBulkCopyColumnMappingCollection();
[C++]
public: IfxBulkCopyColumnMappingCollection();
[JScript]
public function IfxBulkCopyColumnMappingCollection();
```

### Remarks

If you use this constructor to create a IfxBulkCopyColumnMappingCollection instance, you will need to add column mappings. Column mappings can be added by using the Add methods, or the Insert method.

### Example

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined by a IfxBulkCopyColumnMappingCollection instance.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMappingCollection colMapCollection;
  colMapCollection = new IfxBulkCopyColumnMappingCollection();

  salesCopy.ColumnMappings = colMapCollection;

  colMapCollection.Add("DEPTNUMB", "DEPTNO");
  colMapCollection.Add("DEPTNAME", "DEPTNAME");
  colMapCollection.Add("MANAGER", "ADMRDEPT");
  colMapCollection.Add("LOCATION", "LOCATION");

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopyColumnMappingCollection Methods

The methods of the IfxBulkCopyColumnMappingCollection class are listed here.

### Public Methods

| Name | Description |
|---|---|
|  Add | Overloaded. Creates a IfxBulkCopyColumnMapping using column names for the data source and destination columns, and adds it to the IfxBulkCopyColumnMappingCollection. |
|  Clear | Clears the current collection of IfxBulkCopyColumnMapping objects. |
|  Contains | Returns a value indicating if the specified IfxBulkCopyColumnMapping object is in the IfxBulkCopyColumnMappingCollection. |
|  CopyTo | Copies the IfxBulkCopyColumnMapping objects in the IfxBulkCopyColumnMappingCollection to an array of IfxBulkCopyColumnMapping objects, starting at a specified index of the array. |
|  IndexOf | Returns a value indicating the position in the IfxBulkCopyColumnMappingCollection where the specified IfxBulkCopyColumnMapping object is located. |
|  Insert | Inserts the specified IfxBulkCopyColumnMapping to the IfxBulkCopyColumnMappingCollection at the specified index position. |

| Name | Description |
|---|---|
|  Remove | Removes the specified IfxBulkCopyColumnMapping from the IfxBulkCopyColumnMappingCollection. |
|  RemoveAt | Removes the IfxBulkCopyColumnMapping at the specified index position from the IfxBulkCopyColumnMappingCollection. |
|  ToString (Inherited from Object.) | Gets the complete text of the column mappings in the IfxBulkCopyColumnMappingCollection. |

## Protected Methods

| Name | Description |
|---|---|
|  OnInsert (Inherited from CollectionBase.) | Performs additional custom processes before inserting a new mapping into the IfxBulkCopyColumnMappingCollection instance. |

**Reference**

"IfxBulkCopyColumnMappingCollection Members" on page 5-57
"IfxBulkCopyColumnMappingCollection Class" on page 5-56
"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopyColumnMappingCollection.Add Method:**

Creates a IfxBulkCopyColumnMapping using column names for the data source and destination columns, and adds it to the IfxBulkCopyColumnMappingCollection.

**Overload List**

| Method | Description |
|---|---|
| Add(IBM.Data.Informix.IfxBulkCopyColumnMapping) | Adds the specified IfxBulkCopyColumnMapping to the IfxBulkCopyColumnMappingCollection. |
| Add(int, int) | Creates a IfxBulkCopyColumnMapping using column ordinals for the data source and destination columns, and adds it to the IfxBulkCopyColumnMappingCollection. |
| Add(int, string) | Creates a IfxBulkCopyColumnMapping using a column ordinal for the data source column and a column name for the destination column, and adds it to the IfxBulkCopyColumnMappingCollection. |
| Add(string, int) | Creates a IfxBulkCopyColumnMapping using a column name for the data source column and a column ordinal for the destination column, and adds it to the IfxBulkCopyColumnMappingCollection. |
| Add(string, string) | Creates a IfxBulkCopyColumnMapping using column names for the data source and destination columns, and adds it to the IfxBulkCopyColumnMappingCollection. |

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56
"IBM.Data.Informix Namespace" on page 5-1

*IfxBulkCopyColumnMappingCollection.Add (IBM.Data.Informix.IfxBulkCopyColumnMapping) Method:*

Adds the specified IfxBulkCopyColumnMapping to the IfxBulkCopyColumnMappingCollection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function Add ( _
          bulkCopyColumnMapping As IfxBulkCopyColumnMapping _
) As IfxBulkCopyColumnMapping
[C#]
public IfxBulkCopyColumnMapping Add(IfxBulkCopyColumnMapping bulkCopyColumnMapping)
[C++]
public:
IfxBulkCopyColumnMapping* Add (IfxBulkCopyColumnMapping* bulkCopyColumnMapping)
[JScript]
public function Add (
          bulkCopyColumnMapping : IfxBulkCopyColumnMapping
) : IfxBulkCopyColumnMapping
```

**Parameters**

**bulkCopyColumnMapping**
The column mapping to be added to the IfxBulkCopyColumnMappingCollection.

**Return value**

The column mapping added to the IfxBulkCopyColumnMappingCollection.

**Example**

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined by a IfxBulkCopyColumnMappingCollection instance.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMappingCollection colMapCollection;
  colMapCollection = new IfxBulkCopyColumnMappingCollection();

  salesCopy.ColumnMappings = colMapCollection;

  IfxBulkCopyColumnMapping colMapDeptNum =
  new IfxBulkCopyColumnMapping("DEPTNUMB", "DEPTNO");
  IfxBulkCopyColumnMapping colMapDeptNme =
  new IfxBulkCopyColumnMapping("DEPTNAME", "DEPTNAME");
  IfxBulkCopyColumnMapping colMapDeptMgr =
  new IfxBulkCopyColumnMapping("MANAGER", "ADMRDEPT");
  IfxBulkCopyColumnMapping colMapDeptLoc =
  new IfxBulkCopyColumnMapping("LOCATION", "LOCATION");
```

```
  colMapCollection.Add(colMapDeptNum);
  colMapCollection.Add(colMapDeptNme);
  colMapCollection.Add(colMapDeptMgr);
  colMapCollection.Add(colMapDeptLoc);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

*IfxBulkCopyColumnMappingCollection.Add(int, int) Method:*

Creates a IfxBulkCopyColumnMapping using column ordinals for the data source and destination columns, and adds it to the IfxBulkCopyColumnMappingCollection.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function Add ( _
          src As Integer, _
          dest As Integer _
) As IfxBulkCopyColumnMapping
[C#]
public IfxBulkCopyColumnMapping Add(
         int src,
         int dest
)
[C++]
public:
DB2BulkCopyColumnMapping* Add (
         int src,
         int dest
)
[JScript]
public function Add (
          src : int,
          dest : int
) : IfxBulkCopyColumnMapping
```

**Parameters**

**src** Ordinal value of the column being copied from the data source.

**dest** Ordinal value of the column being mapped in the destination table.

**Return value**

The column mapping added to the IfxBulkCopyColumnMappingCollection.

**Example**

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined by a IfxBulkCopyColumnMappingCollection instance.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMappingCollection colMapCollection;
  colMapCollection = new IfxBulkCopyColumnMappingCollection();

  salesCopy.ColumnMappings = colMapCollection;

  colMapCollection.Add(0, 0);
  colMapCollection.Add(1, 1);
  colMapCollection.Add(2, 3);
  colMapCollection.Add(4, 4);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

*IfxBulkCopyColumnMappingCollection.Add(int, string) Method:*

Creates a IfxBulkCopyColumnMapping using a column ordinal for the data source column and a column name for the destination column, and adds it to the IfxBulkCopyColumnMappingCollection.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function Add ( _
          src As Integer, _
          dest As String _
) As IfxBulkCopyColumnMapping
[C#]
public IfxBulkCopyColumnMapping Add(
         int src,
         string dest
)
[C++]
public:
DB2BulkCopyColumnMapping* Add (
         int src,
         String* dest
)
[JScript]
public function Add (
          src : int,
          dest : String
) : IfxBulkCopyColumnMapping
```

### Parameters

**src** Ordinal value of the column being copied from the data source.

**dest** Name of the column being mapped in the destination table.

### Return value

The column mapping added to the IfxBulkCopyColumnMappingCollection.

### Example

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined by a IfxBulkCopyColumnMappingCollection instance.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMappingCollection colMapCollection;
  colMapCollection = new IfxBulkCopyColumnMappingCollection();

  salesCopy.ColumnMappings = colMapCollection;

  colMapCollection.Add(0, "DEPTNO");
  colMapCollection.Add(1, "DEPTNAME");
  colMapCollection.Add(2, "ADMRDEPT");
  colMapCollection.Add(4, "LOCATION");

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
```

```
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

*IfxBulkCopyColumnMappingCollection.Add(string, int) Method:*

Creates a IfxBulkCopyColumnMapping using a column name for the data source column and a column ordinal for the destination column, and adds it to the IfxBulkCopyColumnMappingCollection.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function Add ( _
          src As String, _
          dest As Integer _
) As IfxBulkCopyColumnMapping
[C#]
public IfxBulkCopyColumnMapping Add(
         string src,
         int dest
)
[C++]
public:
DB2BulkCopyColumnMapping* Add (
         String* src,
         int dest
)
[JScript]
public function Add (
          src : String,
          dest : int
) : IfxBulkCopyColumnMapping
```

**Parameters**

**src** Name of the column being copied from the data source.

**dest** Ordinal value of the column being mapped in the destination table.

**Return value**

The column mapping added to the IfxBulkCopyColumnMappingCollection.

### Example

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined by a IfxBulkCopyColumnMappingCollection instance.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMappingCollection colMapCollection;
  colMapCollection = new IfxBulkCopyColumnMappingCollection();

  salesCopy.ColumnMappings = colMapCollection;

  colMapCollection.Add("DEPTNUMB", 0);
  colMapCollection.Add("DEPTNAME", 1);
  colMapCollection.Add("MANAGER", 3);
  colMapCollection.Add("LOCATION", 4);

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

*IfxBulkCopyColumnMappingCollection.Add(string, string) Method:*

Creates a IfxBulkCopyColumnMapping using column names for the data source and destination columns, and adds it to the IfxBulkCopyColumnMappingCollection.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function Add ( _
          src As String, _
          dest As String _
) As IfxBulkCopyColumnMapping
[C#]
public IfxBulkCopyColumnMapping Add(
         string src,
```

```
        string dest
)
[C++]
public:
DB2BulkCopyColumnMapping* Add (
        String* src,
        String* dest
)
[JScript]
public function Add (
         src : String,
         dest : String
) : IfxBulkCopyColumnMapping
```

**Parameters**

**src** Name of the column being copied from the data source.

**dest** Name of the column being mapped in the destination table.

**Return value**

The column mapping added to the IfxBulkCopyColumnMappingCollection.

**Example**

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined by a IfxBulkCopyColumnMappingCollection instance.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMappingCollection colMapCollection;
  colMapCollection = new IfxBulkCopyColumnMappingCollection();

  salesCopy.ColumnMappings = colMapCollection;

  colMapCollection.Add("DEPTNUMB", "DEPTNO");
  colMapCollection.Add("DEPTNAME", "DEPTNAME");
  colMapCollection.Add("MANAGER", "ADMRDEPT");
  colMapCollection.Add("LOCATION", "LOCATION");

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopyColumnMappingCollection.Clear Method:**

Clears the current collection of IfxBulkCopyColumnMapping objects.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub Clear
[C#]
public void Clear ()
[C++]
public:
void Clear ()
[JScript]
public function Clear ()
```

**Remarks**

If you need to perform multiple bulk copies, it would be more efficient to use a single IfxBulkCopy instance than to create a new instance for each bulk copy operation. When performing multiple bulk copy operations, use the Clear method to remove the existing column mappings (after running the IfxBulkCopy.WriteToServer method for the previous operation) before defining new column mappings for the next bulk copy operation.

**Example**

[C#] The following example demonstrates a method that accepts existing IfxBulkCopy and IfxBulkCopyColumnMappingCollection instances and performs bulk copy of data from a IfxDataReader into the DEPARTMENT table. Other methods similar to this one could reuse the IfxBulkCopy and IfxBulkCopyColumnMappingCollection instances, providing they Clear the column mappings before defining their own.

```
[C#]
public static void copyDept(IfxConnection conn, IfxDataReader reader,
                            IfxBulkCopy salesCopy,
                            IfxBulkCopyColumnMappingCollection colMapCollection)
{
  colMapCollection.Clear();
  salesCopy.DestinationTableName = "DEPARTMENT";

  colMapCollection.Add("DEPTNUMB", "DEPTNO");
  colMapCollection.Add("DEPTNAME", "DEPTNAME");
  colMapCollection.Add("MANAGER", "ADMRDEPT");
  colMapCollection.Add("LOCATION", "LOCATION");

  try
  {
    salesCopy.WriteToServer(reader);
    colMapCollection.Clear();
  }
  catch (IfxException ex)
```

```
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopyColumnMappingCollection.Contains Method:**

Returns a value indicating if the specified IfxBulkCopyColumnMapping object is in the IfxBulkCopyColumnMappingCollection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function Contains ( _
          inValue As IfxBulkCopyColumnMapping _
) As Boolean
[C#]
public bool Contains(IfxBulkCopyColumnMapping inValue)
[C++]
public:
bool Contains (IfxBulkCopyColumnMapping* inValue)
[JScript]
public function Contains (
          inValue : IfxBulkCopyColumnMapping
) : boolean
```

**Parameters**

**inValue**
A IfxBulkCopyColumnMapping object.

**Return value**

If the specified IfxBulkCopyColumnMapping exists in the collection, Contains returns `true`. If not, Contains returns `false`.

**Example**

[C#] The following example demonstrates a bulk copy of data from a IfxDataReader into the DEPARTMENT table. The column mappings between the source and target tables are defined by a IfxBulkCopyColumnMappingCollection instance. In addition to the bulk copy operation, the Contains method is demonstrated as it determines if a IfxBulkCopyColumnMapping exists in the collection.

```
[C#]
public static void copyIntoSales(IfxConnection conn, IfxDataReader reader)
{
```

```
  IfxBulkCopy salesCopy = new IfxBulkCopy(conn);
  salesCopy.DestinationTableName = "DEPARTMENT";

  IfxBulkCopyColumnMappingCollection colMapCollection;
  colMapCollection = new IfxBulkCopyColumnMappingCollection();

  salesCopy.ColumnMappings = colMapCollection;

  colMapCollection.Add("DEPTNUMB", "DEPTNO");
  colMapCollection.Add("DEPTNAME", "DEPTNAME");
  colMapCollection.Add("MANAGER", "ADMRDEPT");
  colMapCollection.Add("LOCATION", "LOCATION");

  //Determine if a IfxBulkCopyColumnMapping
  //is included in the current collection.
  IfxBulkCopyColumnMapping colMapDeptNum =
  new IfxBulkCopyColumnMapping("DEPTNUMB", "DEPTNO");

  //If the IfxBulkCopyColumnMapping does exist, print a message indicating this.
  if (colMapCollection.Contains(colMapDeptNum))
  {
    MessageBox.Show(
      "The Department Number mapping is included in the mapping collection.");
  }

  try
  {
    salesCopy.WriteToServer(reader);
    salesCopy.Close();
  }
  catch (IfxException ex)
  {
    MessageBox.Show(ex.ToString(), "Exception");
  }
}
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopyColumnMappingCollection.CopyTo Method:**

Copies the IfxBulkCopyColumnMapping objects in the IfxBulkCopyColumnMappingCollection to an array of IfxBulkCopyColumnMapping objects, starting at a specified index of the array.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub CopyTo ( _
          array As IfxBulkCopyColumnMapping() _
          index As Integer _
)
[C#]
```

```
public void CopyTo(
         IfxBulkCopyColumnMapping[] inValue,
         int index
)
[C++]
public:
void CopyTo (
        array<IfxBulkCopyColumnMapping> array,
        index int
)
[JScript]
public function CopyTo (
          array : IfxBulkCopyColumnMapping[],
          index : int
)
```

**Parameters**

**array** An array that will hold IfxBulkCopyColumnMapping objects that will be copied from the IfxBulkCopyColumnMappingCollection.

**index** An index value indicating where in the array copying will begin.

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

### IfxBulkCopyColumnMappingCollection.IndexOf Method:

Returns a value indicating the position in the IfxBulkCopyColumnMappingCollection where the specified IfxBulkCopyColumnMapping object is located.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function IndexOf ( _
          inValue As IfxBulkCopyColumnMapping _
) As Integer
[C#]
public int IndexOf(IfxBulkCopyColumnMapping inValue)
[C++]
public:
int IndexOf (IfxBulkCopyColumnMapping* inValue)
[JScript]
public function IndexOf (
          inValue : IfxBulkCopyColumnMapping
) : int
```

**Parameters**

**inValue**

The IfxBulkCopyColumnMapping object you are searching for.

**Return value**

The zero-based index of the column mapping. IndexOf will return -1 if the column mapping is not found in the collection.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopyColumnMappingCollection.Insert Method:**

Inserts the specified IfxBulkCopyColumnMapping to the IfxBulkCopyColumnMappingCollection at the specified index position.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub Insert ( _
          index As Integer, _
          inValue As IfxBulkCopyColumnMapping _
)
[C#]
public void Insert(int index, IfxBulkCopyColumnMapping inValue)
[C++]
public:
void Insert (int index, IfxBulkCopyColumnMapping* inValue)
[JScript]
public function Insert (
          index : int,
          inValue : IfxBulkCopyColumnMapping
)
```

**Parameters**

**index** The zero-based index position in the IfxBulkCopyColumnMappingCollection where the IfxBulkCopyColumnMapping object is to be inserted.

**inValue**
The IfxBulkCopyColumnMapping object to be inserted in the IfxBulkCopyColumnMappingCollection.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopyColumnMappingCollection.RemoveAt Method:**

Removes the IfxBulkCopyColumnMapping at the specified index position from the IfxBulkCopyColumnMappingCollection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub RemoveAt ( index As Integer )
[C#]
public void RemoveAt(int index)
[C++]
public:
void RemoveAt (int index)
[JScript]
public function RemoveAt (
          index : int
)
```

**Parameters**

**index** The zero-based index position in the IfxBulkCopyColumnMappingCollection from where the IfxBulkCopyColumnMapping object is to be removed.

**Remarks**

If you need to perform multiple bulk copies, it would be more efficient to use a single IfxBulkCopy instance than to create a new instance for each bulk copy operation. When performing multiple bulk copy operations, use the RemoveAt method to remove selected column mappings (after running the IfxBulkCopy.WriteToServer method for the previous operation) before defining new column mappings for the next bulk copy operation. You can also use the Clear method to remove all existing column mappings.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

**IfxBulkCopyColumnMappingCollection.Remove Method:**

Removes the specified IfxBulkCopyColumnMapping from the IfxBulkCopyColumnMappingCollection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub Remove ( _
          inValue As IfxBulkCopyColumnMapping _
)
[C#]
public void Remove(IfxBulkCopyColumnMapping inValue)
[C++]
public:
void Remove (IfxBulkCopyColumnMapping* inValue)
[JScript]
public function Remove (
          inValue : IfxBulkCopyColumnMapping
)
```

### Parameters

**inValue**
The IfxBulkCopyColumnMapping object to be removed from the IfxBulkCopyColumnMappingCollection.

### Remarks

If you need to perform multiple bulk copies, it would be more efficient to use a single IfxBulkCopy instance than to create a new instance for each bulk copy operation. When performing multiple bulk copy operations, use the Remove method to remove selected column mappings (after running the IfxBulkCopy.WriteToServer method for the previous operation) before defining new column mappings for the next bulk copy operation. You can also use the Clear method to remove all existing column mappings.

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

## IfxBulkCopyOptions Enumeration

Specify options to use with IfxBulkCopy. These options are bit flags, which enable you to combine them in bit masks.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
<Serializable>
Public Enum IfxBulkCopyOptions
[C#]
[Serializable]
public enum IfxBulkCopyOptions
[C++]
[Serializable]
__value public enum IfxBulkCopyOptions
```

```
[JScript]
public
   Serializable
enum IfxBulkCopyOptions
```

### Remarks

You can pass a IfxBulkCopyOptions enumeration into a IfxBulkCopy constructor to define specific behaviors for the WriteToServer methods.

### Members

| Member name | Bit value | Description |
|---|---|---|
| Default | 0 | Uses the default values for all options. |
| KeepIdentity | 1 | Preserves source identity values. Identity columns can be kept only if they are not defined as GENERATED ALWAYS. If they are defined as GENERATED ALWAYS, the WriteToServer method throws an exception.<br><br>When not specified, identity values are assigned by the destination. |
| TableLock | 2 | Assigns a table lock for the duration of the bulk copy operation. Other applications are not permitted to update the table during the copy operation.<br><br>When not specified, table reads are allowed on the table for rows that existed before the copy operation. |
| Truncate | 4 | Clears the data in the destination table before the copy operation starts. |

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxBulkCopy Class" on page 5-19

"IfxBulkCopyColumnMapping Class" on page 5-39

"IfxBulkCopyColumnMappingCollection Class" on page 5-56

"IBM.Data.Informix Namespace" on page 5-1

## IfxClob Class

Represents the CLOB Informix data type. Encapsulates the `string` .NET data type.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
Public Structure IfxClob
[C#]
public struct IfxClob
[C++]
public value class IfxClob
```

## Example

[C#] The following example demonstrates how to retreive a single CLOB column value from a table.

```
[C#]
  public static string getParam(IfxConnection conn)
  {
    string mySelectQuery = "SELECT * FROM EMP_RESUME";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);

    IfxDataReader reader = myCommand.ExecuteReader();

    if (reader.Read())
    {
      IfxClob selectValue = reader.GetIfxClob(2);
      if (!selectValue.IsNull) { return selectValue.ToString(); }
    }

    return "NULL";
  }
```

## Thread safety

Any public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxClob Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxClob Members

Represents the CLOB Informix data type. Encapsulates the string .NET data type. The following tables list the members exposed by the IfxClob class.

### Public Fields

| Field | Description |
|---|---|
|  Null | Null value for IfxClob. |

### Public Constructors

| Constructor | Description |
|---|---|
| IfxClob | Initializes a new IfxClob object with the specified string. |

### Public Properties

| Property | Description |
|---|---|
| CacheData | Indicates if the the data stored in the IfxClob instance's current cursor position is being cached. |
| IsNull | Gets a value that indicates if the value stored in the IfxClob object is null. |
| Value | Gets the value stored in the IfxClob object. |

### Public Methods

| Method | Description |
|---|---|
| GetChars | Reads a stream of characters from the specified column offset into the buffer as an array, starting at the given buffer offset. |
| ToString | Returns a string that represents the IfxClob structure. |

**Reference**

"IfxClob Class" on page 5-75
"IBM.Data.Informix Namespace" on page 5-1

## IfxClob Fields

The fields of the IfxClob structure are listed here.

### Public Fields

| Field | Description |
|---|---|
| Null | Null value for IfxClob. |

**Reference**

"IfxClob Members" on page 5-76
"IfxClob Class" on page 5-75
"IBM.Data.Informix Namespace" on page 5-1

**IfxClob.Null Field:**

Null value for IfxClob.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxClob
[C#]
public static readonly IfxClob Null
[C++]
public:
static initonly IfxClob Null
[JScript]
public static final var Null () : IfxClob
```

**Remarks**

The value of this constant is NULL.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxClob Class" on page 5-75

"IBM.Data.Informix Namespace" on page 5-1

## IfxClob Constructor

Initializes a new IfxClob structure with the specified string.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New(value as String)
[C#]
public IfxClob(string value);
[C++]
public: IfxClob(string value);
[JScript]
public function IfxClob(value : string);
```

### Parameters

**value** A character string to populate the IfxClob instance.

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxClob Class" on page 5-75

"IBM.Data.Informix Namespace" on page 5-1

## IfxClob Methods

The methods of the IfxClob structure are listed here.

## Public Methods

| Method | Description |
|---|---|
|  GetChars | Reads a stream of characters from the specified column offset into the buffer as an array, starting at the given buffer offset. |
|  ToString | Returns a string that represents the IfxClob structure. |

**Reference**

"IfxClob Members" on page 5-76

"IfxClob Class" on page 5-75

"IBM.Data.Informix Namespace" on page 5-1

### IfxClob.GetChars Method:

Reads a stream of characters from the specified column offset into the buffer as an array, starting at the given buffer offset.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetChars( _
   ByVal dataIndex As Long, _
   ByVal buffer() As Char, _
   ByVal bufferIndex As Integer, _
   ByVal length As Integer _
) As Long
[C#]
public long GetChars(
   long dataIndex,
   char[] buffer,
   int bufferIndex,
   int length
);
[C++]
public: __int64 GetChars(
   __int64 dataIndex,
   __wchar_t buffer __gc[],
   int bufferIndex,
   int length
);
[JScript]
public function GetChars(
   dataIndex : long,
   buffer : Char[],
   bufferIndex : int,
   length : int
) : long;
```

### Parameters

**dataIndex**
The index within the row where the read operation is to begin.

**buffer** The buffer into which to copy data.

**bufferIndex**
The index where **buffer** is to begin the write operation.

**length** The number of characters to read.

**Return value**

The actual number of characters read.

**Exceptions**

| Exception type | Condition |
|---|---|
| IfxException | Invalid conversion. |
| InvalidOperationException | There is no more data to return. |

**Remarks**

GetChars returns the number of available characters in the field. In most cases this is the exact length of the field. However, the number returned may be less than the true length of the field if GetChars has already been used to obtain characters from the field. This may be the case, for example, if the IfxClob is reading a CLOB into a buffer. For more information, see the SequentialAccess setting of System.Data.CommandBehavior in the Microsoft(R) .NET Framework SDK documentation.

If you pass a buffer that is a null value. GetChars returns the length of the field in characters.

If the CacheData property is `false`, applications can only retrieve the complete set of data from a IfxClob instance once. Any additional attempts at retrieving data will result in an InvalidOperationException. The specific implications of this are:

- This method can only be used to retrieve the complete set of data from a IfxClob instance once.
- If you use this method to retrieve data from a IfxClob instance, you cannot use the Value property or the ToString method to do the same.
- If you have already used the Value property or the ToString method to retrieve data from a IfxClob instance, you cannot use this method to do the same.

If the CacheData property is `true`, data can be read from the DB2Clob structure, using any of the access methods.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxClob Class" on page 5-75
"IfxClob Members" on page 5-76
"IBM.Data.Informix Namespace" on page 5-1

**IfxClob.ToString Method:**

Returns a string that represents the IfxClob structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxClob structure.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidOperationException | There is no more data to return. |

**Remarks**

If the CacheData property is `false`, applications can only retrieve the complete set of data from a IfxClob instance once. Any additional attempts at retrieving data will result in an InvalidOperationException. The specific implications of this are:

- This method can only be used to retrieve data from a IfxClob instance once.
- If you use this method to retrieve data from a IfxClob instance, you cannot use the Value property or the GetChars method to do the same.
- If you have already used the Value property or the GetChars method to retrieve data from a IfxClob instance, you cannot use this method to do the same.

If the CacheData property is `true`, data can be read from the DB2Clob structure, using any of the access methods.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxClob Class" on page 5-75

"IBM.Data.Informix Namespace" on page 5-1

## IfxClob Properties

The properties of the IfxClob structure are listed here.

## Public Properties

| Property | Description |
|---|---|
| CacheData | Indicates if the the data stored in the IfxClob instance's current cursor position is being cached. |
| IsNull | Gets a value that indicates if the value stored in the IfxClob object is null. |
| Value | Gets the value stored in the IfxClob object. |

**Reference**

"IfxClob Members" on page 5-76
"IfxClob Class" on page 5-75
"IBM.Data.Informix Namespace" on page 5-1

**IfxClob.CacheData Property:**

Indicates if the data stored in the IfxClob instance's current cursor position is being cached.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property CacheData As Boolean
[C#]
public bool CacheData {get;}
[C++]
public: __property bool get_CacheData();
[JScript]
public function get CacheData() : Boolean;
```

**Property value**

`true` if the IfxClob structure's instance is being cached; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxClob Class" on page 5-75
"IfxClob Members" on page 5-76
"IBM.Data.Informix Namespace" on page 5-1

**IfxClob.IsNull Property:**

Gets a value that indicates if the value stored in the IfxClob structure is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxClob Class" on page 5-75

"IBM.Data.Informix Namespace" on page 5-1

**IfxClob.Value Property:**

Gets the value stored in the IfxClob structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Value As String
[C#]
public string Value {get;}
[C++]
public: __property string get_Value();
[JScript]
public function get Value() : string;
```

**Property value**

A character string representing the IfxClob instance.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidOperationException | There is no more data to return. |

**Remarks**

If the CacheData property is `false`, applications can only retrieve the complete set of data from a IfxClob instance once. Any additional attempts at retrieving data will result in an InvalidOperationException. The specific implications of this are:

- This property can only be used to retrieve data from a IfxClob instance once.
- If you use this property to retrieve data from a IfxClob instance, you cannot use the ToString or GetChars methods to do the same.
- If you have already used the ToString or GetChars methods to retrieve data from a IfxClob instance, you cannot use this property to do the same.

If the CacheData property is `true`, data can be read from the DB2Clob structure, using any of the access methods.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxClob Class" on page 5-75

"IBM.Data.Informix Namespace" on page 5-1

# IfxCommandBuilder Class

Automatically generates single-table commands used to reconcile changes made to a DataSet with the associated database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## .NET Framework 2.0 3.0, 3.5, and 4.0 Inheritance hierarchy

System.Object
  System.MarshalByRefObject
    System.ComponentModel.Component
      System.Data.Common.DbCommandBuilder
        IBM.Data.Informix.IfxCommandBuilder

## .NET Framework 2.0 3.0, 3.5, and 4.0 Syntax

```
[Visual Basic]
NotInheritable Public Class IfxCommandBuilder
   Inherits DbCommandBuilder
[C#]
public sealed class IfxCommandBuilder : DbCommandBuilder
[C++]
public __gc __sealed class IfxCommandBuilder : public DbCommandBuilder
[JScript]
public final class IfxCommandBuilder extends DbCommandBuilder
```

## Remarks

The IfxDataAdapter does not automatically generate the SQL statements required to reconcile changes made to a DataSet associated with the database. However, you can create an IfxCommandBuilder object that generates SQL statements for

single-table updates by setting the SelectCommand property of the IfxDataAdapter. Then, the IfxCommandBuilder generates any additional SQL statements that you do not set.

The relationship between a IfxDataAdapter and its corresponding IfxCommandBuilder is always one-to-one. To create this correspondence, you set the DataAdapter property of the IfxCommandBuilder object. This registers the IfxCommandBuilder as a listener, which produces the output of IfxDataAdapter.RowUpdating events that affect the DataSet.

To generate INSERT, UPDATE, or DELETE statements, the IfxCommandBuilder uses the SelectCommand property, which retrieves a required set of metadata. If you change the value of SelectCommand after the metadata has been retrieved (for example, after the first update), you should call the RefreshSchema method to update the metadata.

As you cannot specify a value using the GENERATED ALWAYS AS column, this column will be skipped in the INSERT and SET clause of the UPDATE command.

**Note:** If the SELECT statement assigned to the SelectCommand property uses aliased column names, the resulting INSERT, UPDATE, and DELETE statements might be inaccurate or might fail. If the data server cannot provide the proper base column name for the alias column name, the alias name could be used in the generated INSERT, UPDATE, and DELETE statements. The generated INSERT, UPDATE, and DELETE statements would contain errors.

The IfxCommandBuilder also uses the IfxCommand.Connection, IfxCommand.CommandTimeout, and IfxCommand.Transaction properties referenced by the SelectCommand. The user should call RefreshSchema if any of these properties are modified, or if the value of the SelectCommand property itself is changed. Otherwise, the IfxDataAdapter.InsertCommand, IfxDataAdapter.UpdateCommand, and IfxDataAdapter.DeleteCommand properties retain their previous values.

If you call Dispose, the IfxCommandBuilder is disassociated from the IfxDataAdapter, and the generated commands are no longer used.

The columns that cannot be used in the Where clause (blob, clob, xml) will be skipped from the where clause of the IfxDataAdapter.UpdateCommand, and IfxDataAdapter.DeleteCommand.

The columns that are returned as IsReadOnly=true in IfxDataReader.GetSchemaTable or IsUpdateable=false, will be skipped in the IfxDataAdapter.InsertCommand and set clause of IfxDataAdapter.UpdateCommand.

## Example

[Visual Basic, C#] The following example uses IfxCommand, along with IfxDataAdapter and IfxConnection, to select rows from a database. The example is passed an initialized DataSet, a connection string, a query string that is an SQL SELECT statement, and a string that is the name of the database table. The example then creates an IfxCommandBuilder.

```
[Visual Basic]
Public Function SelectIfxSrvRows(myDataSet As DataSet, myConnection As String,
                mySelectQuery As String, myTableName As String) As DataSet
```

```
    Dim myConn As New IfxConnection(myConnection)
    Dim myDataAdapter As New IfxDataAdapter()
    myDataAdapter.SelectCommand = New IfxCommand(mySelectQuery, myConn)
    Dim employeeCB As IfxCommandBuilder = New IfxCommandBuilder(myDataAdapter)

    myConn.Open()

    Dim employeeDS As DataSet = New DataSet()
    myDataAdapter.Fill(employeeDS, "EMPLOYEE")

    ' Code to modify data in DataSet here

    ' Without the IfxCommandBuilder this line would fail.
    myDataAdapter.Update(employeeDS, "EMPLOYEE")

    myConn.Close()

    Return employeeDS
End Function 'SelectIfxSrvRows
[C#]
public DataSet SelectIfxSrvRows(DataSet
       myDataSet,string myConnection,string mySelectQuery,string myTableName)
{
    IfxConnection myConn = new IfxConnection(myConnection);
    IfxDataAdapter myDataAdapter = new IfxDataAdapter();
    myDataAdapter.SelectCommand = new IfxCommand(mySelectQuery, myConn);
    IfxCommandBuilder employeeCB = new IfxCommandBuilder(myDataAdapter);

    myConn.Open();

    DataSet employeeDS = new DataSet();
    myDataAdapter.Fill(employeeDS, "EMPLOYEE");

    //code to modify data in dataset here

    //Without the IfxCommandBuilder this line would fail
    myDataAdapter.Update(employeeDS, "EMPLOYEE");

    myConn.Close();

    return employeeDS;
 }
```

## Thread safety

Any public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxCommandBuilder Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxCommandBuilder Members

IfxCommandBuilder members

## Public Constructors

| Name | Description |
|---|---|
| IfxCommandBuilder() | Initializes a new instance of the IfxCommandBuilder class. |
| IfxCommandBuilder(IfxDataAdapter) | Initializes a new instance of the IfxCommandBuilder class with the associated IfxDataAdapter object. |

## Public Properties

| Property | Description |
|---|---|
|  CatalogLocation | Gets the CatalogLocation for this IfxCommandBuilder instance. |
|  CatalogSeparator | Gets the string representing a catalog separator for this IfxCommandBuilder instance. |
|  Container (inherited from Component) | Gets the IContainer that contains the Component. |
|  DataAdapter | Gets or sets a IfxDataAdapter object for which this IfxCommandBuilder object will generate SQL statements. |
|  QuotePrefix | Gets or sets the beginning character or characters to use when working with database objects (for example, tables or columns) whose names contain characters such as spaces or reserved tokens. |
|  QuoteSuffix | Gets or sets the ending character or characters to use when working with database objects, (for example, tables or columns), whose names contain characters such as spaces or reserved tokens. |
|  SchemaSeparator | Gets the string representing a schema separator for this IfxCommandBuilder instance. |
|  Site (inherited from Component) | Gets or sets the ISite of the Component. |

## Public Methods

| Method | Description |
|---|---|
|  CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
|   DeriveParameters | Retrieves parameter information from the stored procedure specified in the IfxCommand and populates the Parameters collection of the specified IfxCommand object. |
|  Dispose (inherited from Component) | Overloaded. Releases the resources used by the Component. |

| Method | Description |
|---|---|
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetDeleteCommand | Gets the automatically generated IfxCommand object required to perform deletions at the database. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetInsertCommand | Gets the automatically generated IfxCommand object required to perform insertions at the database. |
| GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| GetUpdateCommand | Gets the automatically generated IfxCommand object required to perform updates at the database. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| QuoteIdentifier | Accepts an unquoted identifier and returns the identifier in a quoted form. |
| RefreshSchema | Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements. |
| UnquoteIdentifier | Accepts a quoted identifier and returns the identifier in an unquoted form. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

## Public Events

| Event | Description |
|---|---|
| Disposed (inherited from Component) | Adds an event handler to listen to the Disposed event on the component. |

## Protected Properties

| Property | Description |
|---|---|
| DesignMode (inherited from Component) | Gets a value that indicates whether the Component is currently in design mode. |
| Events (inherited from Component) | Gets the list of event handlers that are attached to this Component. |

### Protected Methods

| Method | Description |
|---|---|
|  Dispose | Overloaded. Overridden. Releases the resources used by the IfxCommandBuilder. |
|  GetParameterName | Overloaded. Returns the full name of the parameter using the partial parameter name. |
|  GetParameterPlaceholder | Returns the placeholder for the parameter in the associated SQL statement. |
|  GetService (inherited from Component) | Returns an object that represents a service provided by the Component or by its Container. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |
|  SetRowUpdatingHandler | Registers the IfxCommandBuilder to handle the RowUpdating event for a IfxDataAdapter. |

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IBM.Data.Informix Namespace" on page 5-1

## IfxCommandBuilder Constructor

Initializes a new instance of the IfxCommandBuilder class.

### Public Constructors

| Name | Description |
|---|---|
| IfxCommandBuilder() | Initializes a new instance of the IfxCommandBuilder class. |
| IfxCommandBuilder(IfxDataAdapter) | Initializes a new instance of the IfxCommandBuilder class with the associated IfxDataAdapter object. |

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder Constructor ():**

Initializes a new instance of the IfxCommandBuilder class.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New()
[C#]
```

```
public IfxCommandBuilder();
[C++]
public: IfxCommandBuilder();
[JScript]
public function IfxCommandBuilder();
```

**Remarks**

The base constructor initializes all fields to their default values.

**Reference**

"IfxCommandBuilder Constructor" on page 5-89

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder Constructor (IfxDataAdapter):**

Initializes a new instance of the IfxCommandBuilder class with the associated IfxDataAdapter object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New( _
   ByVal adapter As IfxDataAdapter
_
)
[C#]
public IfxCommandBuilder(
   IfxDataAdapter
 adapter
);
[C++]
public: IfxCommandBuilder(
   IfxDataAdapter
* adapter
);
[JScript]
public function IfxCommandBuilder(
   adapter : IfxDataAdapter

);
```

**Parameters**

**adapter**
A IfxDataAdapter object to associate with this IfxCommandBuilder .

**Remarks**

The IfxCommandBuilder registers itself as a listener for IfxDataAdapter.RowUpdating events that are generated by the IfxDataAdapter specified in this property.

When you create a new instance of IfxCommandBuilder, any existing IfxCommandBuilder associated with this IfxDataAdapter is released.

**Reference**

"IfxCommandBuilder Constructor" on page 5-89

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

## IfxCommandBuilder Methods

The methods of the IfxCommandBuilder class are listed here. For a complete list of IfxCommandBuilder class members, see the IfxCommandBuilder Members topic.

### Public Methods

| Method | Description |
|---|---|
| CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
| DeriveParameters | Retrieves parameter information from the stored procedure specified in the IfxCommand and populates the Parameters collection of the specified IfxCommand object. |
| Dispose (inherited from Component) | Overloaded. Releases the resources used by the Component. |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetDeleteCommand | Gets the automatically generated IfxCommand object required to perform deletions at the database. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetInsertCommand | Gets the automatically generated IfxCommand object required to perform insertions at the database. |
| GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| GetUpdateCommand | Gets the automatically generated IfxCommand object required to perform updates at the database. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| QuoteIdentifier | Accepts an unquoted identifier and returns the identifier in a quoted form. |

| Method | Description |
|---|---|
| RefreshSchema | Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements. |
| UnquoteIdentifier | Accepts a quoted identifier and returns the identifier in an unquoted form. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

## Protected Methods

| Method | Description |
|---|---|
| Dispose | Overloaded. Overridden. Releases the resources used by the IfxCommandBuilder. |
| GetParameterName | Overloaded. Returns the full name of the parameter using the partial parameter name. |
| GetParameterPlaceholder | Returns the placeholder for the parameter in the associated SQL statement. |
| GetService (inherited from Component) | Returns an object that represents a service provided by the Component or by its Container. |
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |
| SetRowUpdatingHandler | Registers the IfxCommandBuilder to handle the RowUpdating event for a IfxDataAdapter. |

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder.ApplyParameterInfo Method:**

Overloaded. Assigns a value to the DbParameter.DbType property using the data type from the DataRow. Used internally by DbCommandBuilder.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Sub ApplyParameterInfo( _
   ByVal p As DbParameter, _
   ByVal row As DataRow, _
   ByVal st As StatementType, _
   ByVal whereClause As Boolean _
)
[C#]
public static void ApplyParameterInfo(
   DbParameter p,
```

```
   DataRow row,
   StatementType st,
   boolwhereClause
);
[C++]
public: static void ApplyParameterInfo(
   DbParameter p,
   DataRow row,
   StatementType st,
   boolwhereClause);
[JScript]
public static function ApplyParameterInfo(
   p : DbParameter, _
   row : DataRow, _
   st : StatementType, _
   whereClause : Boolean
);
```

**Parameters**

**p** A DbParameter to which the data type change is applied.

**row** The DataRow from the schema table provided by GetSchemaTable.

**st** The type of command being generated: INSERT, UPDATE or DELETE.

**whereClause**

`true` if the parameter is part of the update or delete WHERE clause.

`false` if it is part of the insert or update values.

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

### IfxCommandBuilder.DeriveParameters Method:

Retrieves parameter information from the stored procedure specified in the IfxCommand and populates the Parameters collection of the specified IfxCommand object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Sub DeriveParameters( _
   ByVal command As IfxCommand

)_
[C#]
public static void DeriveParameters(
   IfxCommand
 command
);
[C++]
public: static void DeriveParameters(
   IfxCommand
* command
);
```

```
[JScript]
public static function DeriveParameters(
   command : IfxCommand

);
```

**Parameters**

**command**

The IfxCommand referencing the stored procedure from which the parameter information is to be derived. The derived parameters are added to the IfxCommand.Parameters collection of the IfxCommand.

**Exceptions**

| Exception type | Condition |
|---|---|
| ArgumentException | |
| InvalidOperationException | The underlying driver does not support returning stored procedure parameter information, or the command text is not a valid stored procedure name, or the CommandType specified was not CommandType.StoredProcedure. |

**Remarks**

DeriveParameters overwrites any existing parameter information for the IfxCommand.

You can only use DeriveParameters with stored procedures. The CommandText must contain the name of the stored procedure and the CommandType must be set to CommandType.StoredProcedure.

Using DeriveParameters on an overloaded stored procedure is not recommended, because all parameters for all qualifying procedures are returned.

By default, DeriveParameters adds the ReturnValue parameter to the Parameters collection of the IfxCommand.

DeriveParameters requires an extra call to the data server to obtain the information. If the parameter information is known in advance, it is more efficient to populate the parameters collection by setting the information explicitly.

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder.Dispose Method:**

Releases the resources used by the IfxCommandBuilder.

### Overload list

| Name | Description |
|---|---|
| Dispose(Boolean) | Releases the unmanaged and, optionally, the managed resources used by the IfxCommandBuilder. |
| Dispose() | Inherited from Component. |

### Example

[Visual Basic, C#] The following example creates a IfxCommandBuilder and then disposes of it.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of Dispose. For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub IfxCommandBuilderHereAndGone()
    Dim myCommandBuilder As New IfxCommandBuilder()
    myCommandBuilder.Dispose()
End Sub

[C#]
public void IfxCommandBuilderHereAndGone() {
   IfxCommandBuilder myCommandBuilder = new IfxCommandBuilder();
   myCommandBuilder.Dispose();
}
```

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

*IfxCommandBuilder.Dispose (Boolean) Method:*

Releases the unmanaged and, optionally, the managed resources used by the IfxCommandBuilder.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Overrides Overloads Protected Sub Dispose( _
   ByVal disposing As Boolean _
)
[C#]
protected override void Dispose(
   bool disposing
);
[C++]
protected: void Dispose(
   bool disposing
);
```

```
[JScript]
protected override function Dispose(
   disposing : Boolean
);
```

**Parameters**

**disposing**
: `true` to release both managed and unmanaged resources; `false` to release only unmanaged resources.

**Remarks**

This method is called by the public Dispose method. Dispose() invokes the protected Dispose(Boolean) method with the **disposing** parameter set to `true`.

When the **disposing** parameter is `true`, the method releases all resources held by any managed objects that this IfxCommand references. It does this by invoking the Dispose() method of each referenced object.

**Example**

[Visual Basic, C#] The following example creates a IfxCommandBuilder and then disposes of it.

```
[Visual Basic]
Public Sub IfxCommandBuilderHereAndGone()
    Dim myCommandBuilder As New IfxCommandBuilder()
    myCommandBuilder.Dispose()
End Sub

[C#]
public void IfxCommandBuilderHereAndGone() {
   IfxCommandBuilder myCommandBuilder = new IfxCommandBuilder();
   myCommandBuilder.Dispose();
}
```

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

"IfxCommandBuilder.Dispose Method" on page 5-94

**IfxCommandBuilder.GetDeleteCommand Method:**

Gets the automatically generated IfxCommand object required to perform deletions at the database.

**Namespace:**
: IBM.Data.Informix

**Assembly:**
: IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetDeleteCommand() As IfxCommand

[C#]
public IfxCommand
 GetDeleteCommand();
```

```
[C++]
public: IfxCommand
* GetDeleteCommand();
[JScript]
public function GetDeleteCommand() : IfxCommand
;
```

**Return value**

The automatically generated IfxCommand object required to perform deletions.

**Remarks**

You can use the GetDeleteCommand method for informational or troubleshooting purposes because it returns the IfxCommand object to be executed. You can also use GetDeleteCommand to delete rows from theIfxDataAdapter.

You can also use GetDeleteCommand as the basis of a modified command. For example, you might call GetDeleteCommand and modify the IfxCommand.CommandTimeout value, and then explicitly set that on the IfxDataAdapter.

After the SQL statement is first generated, you must explicitly call RefreshSchema if it changes the statement in any way. Otherwise, the GetDeleteCommand still will be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either Update or GetDeleteCommand.

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder.GetInsertCommand Method:**

Gets the automatically generated IfxCommand object required to perform insertions at the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetInsertCommand() As IfxCommand

[C#]
public IfxCommand
 GetInsertCommand();
[C++]
public: IfxCommand
* GetInsertCommand();
[JScript]
public function GetInsertCommand() : IfxCommand
;
```

**Return value**

The automatically generated IfxCommand object required to perform insertions.

**Remarks**

You can use the GetInsertCommand method for informational or troubleshooting purposes because it returns the IfxCommand object to be executed. You can also use GetInsertCommand to insert rows in the IfxDataAdapter.

You can also use GetInsertCommand as the basis of a modified command. For example, you might call GetInsertCommand and modify the IfxCommand.CommandTimeout value, and then explicitly set that on the IfxDataAdapter .

After the SQL statement is first generated, you must explicitly call RefreshSchema if it changes the statement in any way. Otherwise, the GetInsertCommand still will be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either Update or GetInsertCommand.

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder.GetParameterName Method:**

Returns the full name of the specified parameter.

**Overload List**

| Method | Description |
|---|---|
| GetParameterName(int) | Returns the name of the parameter specified by the ordinal value. |
| GetParameterName(string) | Returns the full name of the parameter using the partial parameter name. |

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IBM.Data.Informix Namespace" on page 5-1

*IfxCommandBuilder.GetParameterName(int) Method:*

Returns the name of the parameter specified by the ordinal value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function GetParameterName ( _
         parmOrdinal As Integer _
) As String
```

```
[C#]
public override string GetParameterName (int parmOrdinal)
[C++]
public:
virtual String^ GetParameterName (int parmOrdinal) override
[JScript]
public override function GetParameterName (parmOrdinal : int) : String
```

**Parameters**

**parmOrdinal**
An integer representing a column ordinal.

**Return value**

The name of the parameter, with the column ordinal appended to the name.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxCommandBuilder Class" on page 5-84
"IBM.Data.Informix Namespace" on page 5-1

*IfxCommandBuilder.GetParameterName(string) Method:*

Returns the full name of the parameter using the partial parameter name.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function GetParameterName ( _
        parameterName As String _
) As String
[C#]
public override string GetParameterName (string parameterName)
[C++]
public:
virtual String^ GetParameterName (String^ parameterName) override
[JScript]
public override function GetParameterName (parameterName : String) : String
```

**Parameters**

**parameterName**
A string representing a fragment of the parameter name.

**Return value**

The name of the parameter.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IBM.Data.Informix Namespace" on page 5-1

### IfxCommandBuilder.GetParameterPlaceholder Method:

Returns the placeholder for the parameter in the associated SQL statement.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function GetParameterPlaceholder ( _
        parmOrdinal As Integer _
) As String
[C#]
public override string GetParameterPlaceholder (int parmOrdinal)
[C++]
public:
virtual String^ GetParameterPlaceholder (int parmOrdinal) override
[JScript]
public override function GetParameterPlaceholder (parmOrdinal : int) : String
```

**Parameters**

**parmOrdinal**
An integer representing a column ordinal.

**Return value**

The parameter placeholder "?".

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IBM.Data.Informix Namespace" on page 5-1

### IfxCommandBuilder.GetUpdateCommand Method:

Gets the automatically generated IfxCommand object required to perform updates at the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetUpdateCommand() As IfxCommand

[C#]
```

```
public IfxCommand
 GetUpdateCommand();
[C++]
public: IfxCommand
* GetUpdateCommand();
[JScript]
public function GetUpdateCommand() : IfxCommand
;
```

**Return value**

The automatically generated IfxCommand object required to perform updates.

**Remarks**

You can use the GetUpdateCommand method for informational or troubleshooting purposes because it returns the IfxCommand object to be executed. You can also use GetUpdateCommand to update rows in the IfxDataAdapter.

You can also use GetUpdateCommand as the basis of a modified command. For example, you might call GetUpdateCommand and modify the IfxCommand.CommandTimeout value, and then explicitly set that on the IfxDataAdapter .

After the SQL statement is first generated, you must explicitly call RefreshSchema if it changes the statement in any way. Otherwise, the GetUpdateCommand still will be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either Update or GetUpdateCommand.

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder.QuoteIdentifier Method:**

Accepts an unquoted identifier and returns the identifier in a quoted form.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function QuoteIdentifier ( _
   unquotedIdentifier As String _
) As String
[C#]
public override string QuoteIdentifier (
   string unquotedIdentifier
)
[C++]
public:
virtual String^ QuoteIdentifier (
   String^ unquotedIdentifier
) override
```

```
[JScript]
public override function QuoteIdentifier (
   unquotedIdentifier : String
) : String
```

**Parameters**

**unquotedIdentifier**
A String with the identifier in an unquoted form.

**Return value**

A String with the identifier in a quoted form.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommandBuilder Class" on page 5-84
"IfxCommandBuilder Members" on page 5-86
"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder.RefreshSchema Method:**

Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub RefreshSchema()
[C#]
public void RefreshSchema();
[C++]
public: void RefreshSchema();
[JScript]
public function RefreshSchema();
```

**Remarks**

You should call RefreshSchema whenever the IfxDataAdapter.SelectCommand value of the IfxDataAdapter changes.

**Reference**

"IfxCommandBuilder Class" on page 5-84
"IfxCommandBuilder Members" on page 5-86
"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder.SetRowUpdatingHandler Method:**

Registers the IfxCommandBuilder to handle the RowUpdating event for a IfxDataAdapter.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Sub SetRowUpdatingHandler (adapter As DbDataAdapter)
[C#]
public override void SetRowUpdatingHandler (DbDataAdapter adapter)
[C++]
public:
virtual void SetRowUpdatingHandler (DbDataAdapter adapter) override
[JScript]
public override function SetRowUpdatingHandler (adapter : DbDataAdapter)
```

**Parameters**

**adapter**
The DbDataAdapter (or IfxDataAdapter) to be used for the update operation.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder.UnquoteIdentifier Method:**

Accepts a quoted identifier and returns the identifier in an unquoted form.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function UnquoteIdentifier ( _
   quotedIdentifier As String _
) As String
[C#]
public override string UnquoteIdentifier (
   string quotedIdentifier
)
[C++]
public:
virtual String^ UnquoteIdentifier (
   String^ quotedIdentifier
) override
[JScript]
public override function UnquoteIdentifier (
   quotedIdentifier : String
) : String
```

**Parameters**

**unquotedIdentifier**
A String with the identifier in a quoted form.

**Return value**

A String with the identifier in an unquoted form.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommandBuilder Class" on page 5-84
"IfxCommandBuilder Members" on page 5-86
"IBM.Data.Informix Namespace" on page 5-1

## IfxCommandBuilder Properties

The properties of the IfxCommandBuilder class are listed here.

### Public Properties

| Property | Description |
|---|---|
|  CatalogLocation | Gets the CatalogLocation for this IfxCommandBuilder instance. |
|  CatalogSeparator | Gets the string representing a catalog separator for this IfxCommandBuilder instance. |
|  Container (inherited from Component) | Gets the IContainer that contains the Component. |
|  DataAdapter | Gets or sets a IfxDataAdapter object for which this IfxCommandBuilder object will generate SQL statements. |
|  QuotePrefix | Gets or sets the beginning character or characters to use when working with database objects (for example, tables or columns) whose names contain characters such as spaces or reserved tokens. |
|  QuoteSuffix | Gets or sets the ending character or characters to use when working with database objects, (for example, tables or columns), whose names contain characters such as spaces or reserved tokens. |
|  SchemaSeparator | Gets the string representing a schema separator for this IfxCommandBuilder instance. |
|  Site (inherited from Component) | Gets or sets the ISite of the Component. |

## Protected Properties

| Property | Description |
|---|---|
|  DesignMode (inherited from Component) | Gets a value that indicates whether the Component is currently in design mode. |
|  Events (inherited from Component) | Gets the list of event handlers that are attached to this Component. |

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder.CatalogLocation Property:**

Gets the CatalogLocation for this IfxCommandBuilder instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Property CatalogLocation As CatalogLocation
[C#]
public override CatalogLocation CatalogLocation { get; set; }
[C++]
public:
virtual property CatalogLocation CatalogLocation {
 CatalogLocation get () override;
 void set (CatalogLocation value) override;
}
[JScript]
public override function get CatalogLocation () : CatalogLocation
public override function set CatalogLocation (value : CatalogLocation)
```

**Property value**

ReturnsCatalogLocation.Start.

**Exceptions**

| Exception type | Condition |
|---|---|
| System.NotSupportedException | An attempt is made to set the CatalogLocation property. |

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

### IfxCommandBuilder.CatalogSeparator Property:

Gets the string representing a catalog separator for this IfxCommandBuilder instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

#### Syntax

```
[Visual Basic]
Public Overrides Property CatalogSeparator As String
[C#]
public override string CatalogSeparator { get; set; }
[C++]
public:
virtual property String^ CatalogSeparator {
   String^ get () override;
   void set (String^ value) override;
}
[JScript]
public override function get CatalogSeparator () : String
public override function set CatalogSeparator (value : String)
```

#### Property value

The string value for the CatalogSeparator property is ".". For Informix, the string value is ":".

#### Exceptions

| Exception type | Condition |
|---|---|
| System.NotSupportedException | An attempt is made to set the CatalogSeparator property. |

#### Version information

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommandBuilder Class" on page 5-84
"IfxCommandBuilder Members" on page 5-86
"IBM.Data.Informix Namespace" on page 5-1

### IfxCommandBuilder.DataAdapter Property:

Gets or sets a IfxDataAdapter object for which this IfxCommandBuilder object will generate SQL statements.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property DataAdapter As IfxDataAdapter

[C#]
public IfxDataAdapter
 DataAdapter {get; set;}
[C++]
public: __property IfxDataAdapter
* get_DataAdapter();
public: __property void set_DataAdapter(IfxDataAdapter
*);
[JScript]
public function get DataAdapter() : IfxDataAdapter
;
public function set DataAdapter(IfxDataAdapter
);
```

**Property value**

A IfxDataAdapter object that is associated with this IfxCommandBuilder .

**Remarks**

The IfxCommandBuilder registers itself as a listener for IfxDataAdapter.RowUpdating events that are generated by the IfxDataAdapter specified in this property.

When you create a new instance of IfxCommandBuilder, any existing IfxCommandBuilder associated with this IfxDataAdapter is released.

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder.QuotePrefix Property:**

Gets or sets the beginning character or characters to use when working with database objects (for example, tables or columns) whose names contain characters such as spaces or reserved tokens.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property QuotePrefix As String
[C#]
public string QuotePrefix {get; set;}
[C++]
public: __property String* get_QuotePrefix();
public: __property void set_QuotePrefix(String*);
[JScript]
public function get QuotePrefix() : String;
public function set QuotePrefix(String);
```

**Property value**

The beginning character or characters to use. The default is an empty string.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidOperationException | This property cannot be changed after an insert, update, or delete command has been generated. |

**Remarks**

Some databases may contain objects whose names include characters such as spaces, commas, and semicolons. To accommodate this, use the QuotePrefix and QuoteSuffix properties to specify delimiters, such as a left and right bracket, that will encapsulate the object name.

**Note:** Although you cannot change the QuotePrefix or QuoteSuffix properties after an insert, update, or delete command has been generated, you can change their settings after calling the Update method of a IfxDataAdapter.

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommandBuilder.QuoteSuffix Property:**

Gets or sets the ending character or characters to use when working with database objects, (for example, tables or columns), whose names contain characters such as spaces or reserved tokens.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property QuoteSuffix As String
[C#]
public string QuoteSuffix {get; set;}
[C++]
public: __property String* get_QuoteSuffix();
public: __property void set_QuoteSuffix(String*);
[JScript]
public function get QuoteSuffix() : String;
public function set QuoteSuffix(String);
```

**Property value**

The ending character or characters to use. The default is an empty string.

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidOperationException | This property cannot be changed after an insert, update, or delete command has been generated. |

### Remarks

Some databases may contain objects whose names include characters such as spaces, commas, and semicolons. To accommodate this, use the QuotePrefix and QuoteSuffix properties to specify delimiters, such as a left and right bracket, that will encapsulate the object name.

**Note:** Although you cannot change the QuotePrefix or QuoteSuffix properties after an insert, update, or delete operation has been generated, you can change their settings after calling the Update method of a IfxDataAdapter.

**Reference**

"IfxCommandBuilder Class" on page 5-84

"IfxCommandBuilder Members" on page 5-86

"IBM.Data.Informix Namespace" on page 5-1

## IfxCommandBuilder.SchemaSeparator Property:

Gets the string representing a schema separator for this IfxCommandBuilder instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Overrides Property SchemaSeparator As String
[C#]
public override string SchemaSeparator { get; set; }
[C++]
public:
virtual property String^ SchemaSeparator {
 String^ get () override;
 void set (String^ value) override;
}
[JScript]
public override function get SchemaSeparator () : String
public override function set SchemaSeparator (value : String)
```

### Property value

The string value for the SchemaSeparator property is always ".".

### Exceptions

| Exception type | Condition |
|---|---|
| System.NotSupportedException | An attempt is made to set the SchemaSeparator property. |

**Version information**

**.NET Framework version**
Supported in: 2.0

**Reference**

"IfxCommandBuilder Class" on page 5-84
"IfxCommandBuilder Members" on page 5-86
"IBM.Data.Informix Namespace" on page 5-1

# IfxCommand Class

Represents an SQL statement or stored procedure to execute against a data source.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## .NET Framework 2.0 3.0, 3.5, and 4.0 Inheritance hierarchy

System.Object
  System.MarshalByRefObject
    System.ComponentModel.Component
      System.Data.Common.DbCommand
        IBM.Data.Informix.IfxCommand

## .NET Framework 2.0 3.0, 3.5, and 4.0 Syntax

```
[Visual Basic]
NotInheritable Public Class IfxCommand
   Inherits DbCommand
   Implements ICloneable
[C#]
public sealed class IfxCommand : DbCommand, ICloneable
[C++]
public __gc __sealed class IfxCommand : public DbCommand,
   ICloneable
[JScript]
public class IfxCommand extends DbCommand implements ICloneable
```

## Remarks

The IfxCommand class provides the following methods for executing commands against a database:

| Item | Description |
|---|---|
| ExecuteReader | Executes commands that return rows. |
| ExecuteNonQuery | Executes commands such as SQL INSERT, DELETE, UPDATE, and SET statements. |
| ExecuteScalar | Retrieves a single value (for example, an aggregate value) from a database. |

If execution of the command results in a fatal IfxException , the IfxConnection may close. However, the user can reopen the connection and continue.

## Example

[Visual Basic, C#] The following example uses the ExecuteReader method of IfxCommand, along with IfxDataReader and IfxConnection , to select rows from a table.

```
[Visual Basic]
Public Sub ReadMyData(myConnString As String)
     Dim mySelectQuery As String = "SELECT SALES, SALES_PERSON FROM SALES"
     Dim myConnection As New IfxConnection(myConnString)
     Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
     myConnection.Open()
     Dim myReader As IfxDataReader = myCommand.ExecuteReader()
     Try
         While myReader.Read()
             Console.WriteLine(myReader.GetInt32(0).ToString() + ", " _
                + myReader.GetString(1))
         End While
     Finally
         ' always call Close when done reading.
         myReader.Close()
         ' always call Close when done with connection.
         myConnection.Close()
     End Try
 End Sub
```

```
[C#]
public void ReadMyData(string myConnString)
{
    string mySelectQuery = "SELECT SALES, SALES_PERSON FROM SALES";
    IfxConnection myConnection = new IfxConnection(myConnString);
    IfxCommand myCommand = new IfxCommand(mySelectQuery,myConnection);
    myConnection.Open();
    IfxDataReader myReader = myCommand.ExecuteReader();
    try
    {
       while (myReader.Read())
       {
          Console.WriteLine(myReader.GetInt32(0) + ", " + myReader.GetString(1));
       }
    }
    finally
    {
       // always call Close when done reading.
       myReader.Close();
       // always call Close when done with connection.
       myConnection.Close();
    }
}
```

## Thread safety

Any public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

"IfxDataAdapter Class" on page 5-253

"IfxConnection Class" on page 5-157

## IfxCommand Members

IfxCommand overview

### Public Constructors

| Name | Description |
|---|---|
| IfxCommand() | Initializes a new instance of the IfxCommand class. |
| IfxCommand(String) | Initializes a new instance of the IfxCommand class with the text of the query. |
| IfxCommand(String, IfxConnection) | Initializes a new instance of the IfxCommand class with the text of the query and a DB2Connection object. |
| IfxCommand(String, IfxConnection, IfxTransaction) | Initializes a new instance of the IfxCommand class with the text of the query, a DB2Connection object, and the Transaction. |

### Public Properties

| Property | Description |
|---|---|
| CommandText | Gets or sets the SQL statement, or stored procedure to run against the database. |
| CommandTimeout | Gets or sets the wait time before terminating the execution of a command and generating an error. |
| CommandType | Gets or sets a value indicating how the CommandText property is interpreted. |
| Connection | Gets or sets the IfxConnection that is used by an instance of the IfxCommand. |
| Container (inherited from Component) | Gets the IContainer that contains the Component. |
| IfxTypeOutput | Determines whether the output-only parameter values associated with the IfxCommand are returned as native DB2 data types (specifically, classes and structures in the IBM.Data.IfxTypes namespace). |
| DbConnection | Gets or sets the DbConnection used by an instance of the IfxCommand. |
| DbParameterCollection | Gets the DbParameterCollection object. |
| DbTransaction | Gets or sets the DbTransaction within which the IfxCommand runs. |
| DesignTimeVisible | Gets or sets a boolean value indicating whether an IfxCommand generated by a data adapter is visible. |
| DisableCursorHold | Controls the effect of a transaction completion on open cursors. |

| Property | Description |
|---|---|
|  EnableExtendedIndicators | Enables the use of default and unassigned indicators as parameters. |
|  FitHighPrecisionType | Gets or sets the type of precision to be used. The acceptable values are as follows:<br>• `WithTruncate`. Results in returning a .NET system type after silently truncating the column value if needed.<br>• `AsString`. Results in converting the column to a .NET string type.<br>• `ReturnException`. Results in a truncation exception if the value does not fit in the .NET system type. |
|  Parameters | Gets the IfxParameterCollection . |
|  ResultSetAsReturnValue | Set this value to retrieve result sets from Informix user-defined routines (UDR) as a ReturnValue parameter. This property is ignored when a ReturnValue parameter is not bound in the parameter collection. |
|  Site (inherited from Component) | Gets or sets the ISite of the Component. |
|  StatementConcentrator | Gets or sets the value determining whether or not to enable statement concentrator literals. |
|  Transaction | Gets or sets the IfxTransaction within which the IfxCommand runs. |
|  UpdatedRowSource | Gets or sets a value that specifies how the Update method applies command results to the DataRow. |

## Public Methods

| Method | Description |
|---|---|
|  Cancel | Attempts to cancel the execution of a IfxCommand. |
|  CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
|  CreateParameter | Creates a new instance of a IfxParameter object. |
|  Dispose (inherited from Component) | Overridden. Releases the resources used by the Component. |
|  Equals (inherited from Object) | Overridden. Determines whether two Object instances are equal. |
|  ExecuteNonQuery | Executes an SQL statement against the Connection and returns the number of rows affected. |

| Method | Description |
|---|---|
|  ExecutePageReader | Returns a IfxDataReader instance containing a requested set of rows. |
|  ExecuteReader | Overloaded. Sends the CommandText to the Connection and builds a IfxDataReader . |
|  ExecuteScalar | Executes the query, and returns the first column of the first row in the resultset returned by the query. Extra columns or rows are ignored. |
|  ExecuteRow | Sends the CommandText to the Connection and builds a IfxRecord. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
|  GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
|  Prepare | Creates a prepared (or compiled) version of the command at the database. |
|  ResetCommandTimeout | Resets the CommandTimeout property to the default value. |
|  ToString (inherited from Object) | Returns a String that represents the current Object. |

## Public Events

| Event | Description |
|---|---|
|  Disposed (inherited from Component) | Adds an event handler to listen to the Disposed event on the component. |

## Protected Properties

| Property | Description |
|---|---|
|  DesignMode (inherited from Component) | Gets a value that indicates whether the Component is in design mode. |
|  Events (inherited from Component) | Gets the list of event handlers that are attached to a Component. |

### Protected Methods

| Method | Description |
|---|---|
|  CreateDbParameter | Creates a new instance of a DbParameter object. |
|  Dispose | Overloaded. Overridden. Releases the resources used by the IfxCommand. |
|  ExecuteDbDataReader | Sends the CommandText to the Connection and builds a DbDataReader object. |
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  GetService (inherited from Component) | Returns an object that represents a service provided by the Component or by its Container. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxCommand Class" on page 5-110

"IBM.Data.Informix Namespace" on page 5-1

## IfxCommand Constructor

Initializes a new instance of the IfxCommand class.

### Public Constructors

| Name | Description |
|---|---|
| IfxCommand() | Initializes a new instance of the IfxCommand class. |
| IfxCommand(String) | Initializes a new instance of the IfxCommand class with the text of the query. |
| IfxCommand(String, IfxConnection) | Initializes a new instance of the IfxCommand class with the text of the query and a DB2Connection object. |
| IfxCommand(String, IfxConnection, IfxTransaction) | Initializes a new instance of the IfxCommand class with the text of the query, a DB2Connection object, and the Transaction. |

### Example

[Visual Basic, C#] The following example creates a IfxCommand and sets some of its properties.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of the IfxCommand constructor. For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub CreateMyIfxCommand()
    Dim myConnection As New IfxConnection _
       ("DATABASE=SAMPLE")
    myConnection.Open()
    Dim myTrans As IfxTransaction = myConnection.BeginTransaction()
    Dim mySelectQuery As String = _
       "SELECT * FROM EMPLOYEE ORDER BY EMPNO"
    Dim myCommand As New IfxCommand(mySelectQuery, myConnection, myTrans)
    myCommand.CommandTimeout = 20
End Sub

[C#]
public void CreateMyIfxCommand()
{
   IfxConnection myConnection = new IfxConnection("DATABASE=SAMPLE");
   myConnection.Open();
   IfxTransaction myTrans = myConnection.BeginTransaction();
   string mySelectQuery = "SELECT * FROM EMPLOYEE ORDER BY EMPNO";
   IfxCommand myCommand = new IfxCommand(mySelectQuery, myConnection,myTrans);
   myCommand.CommandTimeout = 20;
}
```

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand Constructor ():**

Initializes a new instance of the IfxCommand class.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New()
[C#]
public IfxCommand();
[C++]
public: IfxCommand();
[JScript]
public function IfxCommand();
```

**Remarks**

The base constructor initializes all fields to their default values. The following table shows initial property values for an instance of IfxCommand .

| Properties | Initial value |
|---|---|
| CommandText | empty string ("") |
| CommandTimeout | 30 |
| CommandType | Text |

| Properties | Initial value |
|---|---|
| Connection | null |

### Example

[Visual Basic, C#] The following example creates a IfxCommand and sets some of its properties.

```
[Visual Basic]
Public Sub CreateMyIfxCommand()
    Dim myCommand As New IfxCommand()
    myCommand.CommandTimeout = 20
End Sub

[C#]
public void CreateMyIfxCommand()
{
   IfxCommand myCommand = new IfxCommand();
   myCommand.CommandTimeout = 20;
}
```

### Reference


"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

"IfxCommand Constructor" on page 5-115

"IfxDataAdapter Class" on page 5-253

"IfxConnection Class" on page 5-157


## IfxCommand Constructor (String):

Initializes a new instance of the IfxCommand class with the text of the query.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New( _
   ByVal cmdText As String _
)
[C#]
public IfxCommand(
   string cmdText
);
[C++]
public: IfxCommand(
   String* cmdText
);
[JScript]
public function IfxCommand(
   cmdText : String
);
```

**Parameters**

**cmdText**
The text of the query.

**Example**

[Visual Basic, C#] The following example creates a IfxCommand and sets some of its properties.

```
[Visual Basic]
Public Sub CreateMyIfxCommand()
    Dim mySelectQuery As String = "SELECT * FROM EMPLOYEE ORDER BY EMPNO"
    Dim myCommand As New IfxCommand(mySelectQuery)
    myCommand.CommandTimeout = 20
End Sub

[C#]
public void CreateMyIfxCommand()
{
   string mySelectQuery = "SELECT * FROM EMPLOYEE ORDER BY EMPNO";
   IfxCommand myCommand = new IfxCommand(mySelectQuery);
   myCommand.CommandTimeout = 20;
}
```

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

"IfxCommand Constructor" on page 5-115

**IfxCommand Constructor (String, IfxConnection):**

Initializes a new instance of the IfxCommand class with the text of the query and a IfxConnection object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New( _
   ByVal cmdText As String, _
   ByVal connection As IfxConnection _
)
[C#]
public IfxCommand(
   string cmdText,
   IfxConnection connection
);
[C++]
public: IfxCommand(
   String* cmdText,
   IfxConnection * connection
);
[JScript]
public function IfxCommand(
   cmdText : String,
   connection : IfxConnection
);
```

**Parameters**

**cmdText**
The text of the query.

**connection**
A IfxConnection object that represents a connection to a data server.

**Example**

[Visual Basic, C#] The following example creates a IfxCommand and sets some of its properties.

```
[Visual Basic]
Public Sub CreateMyIfxCommand()
     Dim myConnection As New IfxConnection _
        ("DATABASE=SAMPLE;")
     Dim mySelectQuery As String = _
        "SELECT * FROM EMPLOYEE ORDER BY EMPNO"
     Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
     myCommand.CommandTimeout = 20
 End Sub

[C#]
public void CreateMyIfxCommand()
 {
    IfxConnection myConnection = new IfxConnection("DATABASE=SAMPLE;");
    string mySelectQuery = "SELECT * FROM EMPLOYEE ORDER BY EMPNO";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, myConnection);
    myCommand.CommandTimeout = 20;
 }
```

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

"IfxCommand Constructor" on page 5-115

**IfxCommand Constructor (String, IfxConnection, IfxTransaction):**

Initializes a new instance of the IfxCommand class with the text of the query, a IfxConnection object, and the Transaction.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New( _
   ByVal cmdText As String, _
   ByVal connection As IfxConnection , _
   ByVal transaction As IfxTransaction _
)
[C#]
public IfxCommand(
   string cmdText,
   IfxConnection connection,
   IfxTransaction transaction
);
[C++]
```

```
public: IfxCommand(
   String* cmdText,
   IfxConnection* connection,
   IfxTransaction* transaction
);
[JScript]
public function IfxCommand(
   cmdText : String,
   connection : IfxConnection,
   transaction : IfxTransaction
);
```

### Parameters

**cmdText**
The text of the query.

**connection**
A IfxConnection object that represents an open connection to a data server.

**transaction**
The transaction in which the DB2Command executes.

### Remarks

The following table shows initial property values for an instance of this implementation of the IfxCommand.

| Properties | Initial value |
|---|---|
| CommandText | **cmdText** |
| CommandTimeout | 30 |
| CommandType | Text |
| Connection | A new IfxConnection that is the value for the **connection** parameter. |

You can change the value for any of these parameters by setting the related property.

### Example

[Visual Basic, C#] The following example creates a IfxCommand and sets some of its properties.

```
[Visual Basic]
Public Sub CreateMyIfxCommand()
    Dim myConnection As New IfxConnection _
       ("DATABASE=SAMPLE;")
    myConnection.Open()
    Dim myTrans As IfxTransaction = myConnection.BeginTransaction()
    Dim mySelectQuery As String = _
       "SELECT * FROM EMPLOYEE ORDER BY EMPNO"
    Dim myCommand As New IfxCommand(mySelectQuery, myConnection, myTrans)
    myCommand.CommandTimeout = 20
End Sub

[C#]
public void CreateMyIfxCommand()
{
   IfxConnection myConnection = new IfxConnection("DATABASE=SAMPLE;");
   myConnection.Open();
   IfxTransaction myTrans = myConnection.BeginTransaction();
```

```
   string mySelectQuery = "SELECT * FROM EMPLOYEE ORDER BY EMPNO";
   IfxCommand myCommand = new IfxCommand(mySelectQuery, myConnection,myTrans);
   myCommand.CommandTimeout = 20;
}
```

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

"IfxCommand Constructor" on page 5-115

## IfxCommand Methods

The methods of the IfxCommand class are listed here. For a complete list of IfxCommand class members, see the IfxCommand Members topic.

### Public Methods

| Method | Description |
|---|---|
|  Cancel | Attempts to cancel the execution of a IfxCommand. |
|  CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
|  CreateParameter | Creates a new instance of a IfxParameter object. |
|  Dispose (inherited from Component) | Overridden. Releases the resources used by the Component. |
|  Equals (inherited from Object) | Overridden. Determines whether two Object instances are equal. |
|  ExecuteNonQuery | Executes an SQL statement against the Connection and returns the number of rows affected. |
|  ExecutePageReader | Returns a IfxDataReader instance containing a requested set of rows. |
|  ExecuteReader | Overloaded. Sends the CommandText to the Connection and builds a IfxDataReader . |
|  ExecuteScalar | Executes the query, and returns the first column of the first row in the resultset returned by the query. Extra columns or rows are ignored. |
|  ExecuteRow | Sends the CommandText to the Connection and builds a IfxRecord. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
|  GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |

| Method | Description |
|---|---|
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| Prepare | Creates a prepared (or compiled) version of the command at the database. |
| ResetCommandTimeout | Resets the CommandTimeout property to the default value. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

## Protected Methods

| Method | Description |
|---|---|
| CreateDbParameter | Creates a new instance of a DbParameter object. |
| Dispose | Overloaded. Overridden. Releases the resources used by the IfxCommand. |
| ExecuteDbDataReader | Sends the CommandText to the Connection and builds a DbDataReader object. |
| Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
| GetService (inherited from Component) | Returns an object that represents a service provided by the Component or by its Container. |
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxCommand Class" on page 5-110

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.Cancel Method:**

Attempts to cancel the execution of a IfxCommand.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Public Sub Cancel()
[C#]
```

```
public void Cancel();
[C++]
public: __sealed void Cancel();
[JScript]
public function Cancel();
```

**Remarks**

If there is nothing to cancel, nothing happens. However, if there is a command in process, and the attempt to cancel fails, no exception is generated.

**Example**

[Visual Basic, C#] The following example creates a IfxCommand , executes it, then cancels the execution. To accomplish this, the method is passed a string that is an SQL SELECT statement and a string to use to connect to the database.

```
[Visual Basic]
Public Sub CreateMyIfxCommand _
(mySelectQuery As String, myConnectionString As String)
    Dim myConnection As New IfxConnection(myConnectionString)
    Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
    myCommand.Connection.Open()
    myCommand.ExecuteReader()
    myCommand.Cancel()
End Sub

[C#]
public void CreateMyIfxCommand(string mySelectQuery, string myConnectionString)
{
   IfxConnection myConnection = new IfxConnection(myConnectionString);
   IfxCommand myCommand = new IfxCommand(mySelectQuery, myConnection);
   myCommand.Connection.Open();
   myCommand.ExecuteReader();
   myCommand.Cancel();
}
```

**Note:** DB2Command.Cancel will not close the cursor associated with the reader object (such as DB2DataReader or DB2ResultSet) created by using the DB2Command object. The application need to call the Close method on the reader object in order to release database resources associated with the reader object.

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.CreateDbParameter Method:**

Creates a new instance of a DbParameter object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overrides Protected Function CreateDbParameter() _
   As DbParameter
[C#]
```

```
protected override DbParameter CreateDbParameter();
[C++]
protected: DbParameter CreateDbParameter();
[JScript]
protected override function CreateDbParameter() : DbParameter;
```

**Return value**

A DbParameter instance.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommand Class" on page 5-110
"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.CreateParameter Method:**

Creates a new instance of a IfxParameter object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function CreateParameter() As IfxParameter

[C#]
public IfxParameter
 CreateParameter();
[C++]
public: IfxParameter
* CreateParameter();
[JScript]
public function CreateParameter() : IfxParameter
;
```

**Return value**

A IfxParameter object.

**Reference**

"IfxCommand Class" on page 5-110
"IfxCommand Members" on page 5-112
"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.Dispose Method:**

Releases the resources used by the IfxCommand.

### Overload list

| Name | Description |
|---|---|
| Dispose(Boolean) | Releases the unmanaged and, optionally, the managed resources used by the DB2Command object. |
| Dispose() | Inherited from Component. |

### Example

[Visual Basic, C#] The following example creates a IfxCommand and then disposes of it.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of Dispose. For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub IfxCommandHereAndGone()
    Dim myCommand As New IfxCommand()
    myCommand.Dispose()
End Sub

[C#]
public void IfxCommandHereAndGone()
{
   IfxCommand myCommand = new IfxCommand();
   myCommand.Dispose();
}
```

#### Reference

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

*IfxCommand.Dispose (Boolean) Method:*

Releases the unmanaged and, optionally, the managed resources used by the IfxCommand object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Overrides Overloads Protected Sub Dispose( _
   ByVal disposing As Boolean _
)
[C#]
protected override void Dispose(
   bool disposing
);
[C++]
protected: void Dispose(
   bool disposing
);
```

```
[JScript]
protected override function Dispose(
   disposing : Boolean
);
```

**Parameters**

**disposing**
: `true` to release both managed and unmanaged resources; `false` to release only unmanaged resources.

**Remarks**

This method is called by the public Dispose method. Dispose() invokes the protected Dispose(Boolean) method with the **disposing** parameter set to `true`.

When the **disposing** parameter is `true`, the method releases all resources held by any managed objects that this IfxCommand references. It does this by invoking the Dispose() method of each referenced object.

**Notes to inheritors:**

Dispose can be called multiple times by other objects. When overriding Dispose(Boolean), be careful not to reference objects that have been previously disposed of in an earlier call to Dispose. For more information about how to implement Dispose(Boolean), see "Implementing a Dispose Method" in the Microsoft(R) .NET Framework SDK documentation.

**Example**

[Visual Basic, C#] The following example creates a IfxCommand and then disposes of it.

```
[Visual Basic]
Public Sub IfxCommandHereAndGone()
    Dim myCommand As New IfxCommand()
    myCommand.Dispose()
End Sub

[C#]
public void IfxCommandHereAndGone()
{
   IfxCommand myCommand = new IfxCommand();
   myCommand.Dispose();
}
```

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

"IfxCommand.Dispose Method" on page 5-124

**IfxCommand.ExecuteDbDataReader Method:**

Sends the CommandText to the Connection and builds a DbDataReader object.

**Namespace:**
: IBM.Data.Informix

**Assembly:**
: IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Overrides Protected Function ExecuteDbDataReader( _
   ByVal behavior As CommandBehavior _
) As DbDataReader
[C#]
protected override DbDataReader ExecuteDbDataReader(
   CommandBehavior behavior
);
[C++]
protected: DbDataReader ExecuteDbDataReader(
   CommandBehavior behavior
);
[JScript]
protected override function ExecuteDbDataReader(
   behavior : CommandBehavior
) : DbDataReader;
```

### Parameters

**behavior**
A CommandBehavior instance.

### Version information

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommand Class" on page 5-110

"IBM.Data.Informix Namespace" on page 5-1

## IfxCommand.ExecuteNonQuery Method:

Executes an SQL statement against the Connection and returns the number of rows affected.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
NotOverridable Public Function ExecuteNonQuery() As Integer
[C#]
public int ExecuteNonQuery();
[C++]
public: __sealed int ExecuteNonQuery();
[JScript]
public function ExecuteNonQuery() : int;
```

### Return value

For UPDATE, INSERT, and DELETE statements, the return value is the number of rows affected by the command. For all other types of statements, the return value is -1.

**Remarks**

You can use ExecuteNonQuery to perform catalog operations (for example, querying the structure of a database or creating database objects such as tables); or to change the data in a database, without using a DataSet, by executing UPDATE, INSERT, or DELETE statements.

You can also use ExecuteNonQuery to execute multiple SQL statements. In this case, the return value is the number of rows affected by all statements in the command.

Although ExecuteNonQuery does not return any rows, any output parameters or return values mapped to parameters are populated with data.

**Example**

[Visual Basic, C#] The following example creates a IfxCommand and then executes it by using ExecuteNonQuery. The example is passed a string that is an SQL statement (such as UPDATE, INSERT, or DELETE) and a string to use to connect to the database.

```
[Visual Basic]
Public Sub CreateMyIfxCommand(myExecuteQuery As String, _
myConnectionString As String)
    Dim myConnection As New IfxConnection(myConnectionString)
    Dim myCommand As New IfxCommand(myExecuteQuery, myConnection)
    myCommand.Connection.Open()
    myCommand.ExecuteNonQuery()
    MyConnection.Close()
End Sub

[C#]
public void CreateMyIfxCommand(string myExecuteQuery, string myConnectionString)
{
   IfxConnection myConnection = new IfxConnection(myConnectionString);
   IfxCommand myCommand = new IfxCommand(myExecuteQuery, myConnection);
   myCommand.Connection.Open();
   myCommand.ExecuteNonQuery();
   myConnection.Close();
}
```

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.ExecutePageReader Method:**

Returns a IfxDataReader instance containing a requested set of rows.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function ExecutePageReader( _
   ByVal startRow As int, ByVal numRows As int _
) As DataReader
```

```
[C#]
public DataReader ExecutePageReader( int startRow, int numRows);
[C++]
public: DataReader^ ExecutePageReader(
   int startRow, int numRows);
[JScript]
public function ExecutePageReader (
   startRow : int, numRows : int
) : DataReader
```

**Parameters**

**startRow**
Specifies the starting row of the set of rows to be returned.

**numRows**
Specifies total number of rows to be returned.

**Return value**

Returns a IfxDataReader instance containing a requested set of rows.

**Example**

[C#] The following example demonstrates the retrieval of a specific set of result set rows. In the example, a IfxDataReader containing rows three through seven is passed back to the application.

```
[C#]
    IfxCommand cmd = conn.CreateCommand();
    cmd.CommandText = "SELECT * FROM SALES";
    IfxDataReader dr = cmd.ExecutePageReader(3, 7);
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.ExecuteReader Method:**

Sends the CommandText to the Connection and builds a IfxDataReader.

**Overload list**

| Name | Description |
|---|---|
| ExecuteReader() | Sends the CommandText to the Connection and builds a IfxDataReader. |
| ExecuteReader(CommandBehavior) | Sends the CommandText to the Connection, and builds a IfxDataReader based on the CommandBehavior specified. |

### Example

[Visual Basic, C#] The following example creates a IfxCommand, then executes it by passing a string that is an SQL SELECT statement, and a string to use to connect to the database. CommandBehavior is then set to CloseConnection.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of ExecuteReader. For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub CreateMyIfxDataReader(mySelectQuery As String, _
myConnectionString As String)
    Dim myConnection As New IfxConnection(myConnectionString)
    Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
    myCommand.Connection.Open()
    Dim myReader As IfxDataReader =
        myCommand.ExecuteReader(CommandBehavior.CloseConnection)
    While myReader.Read()
        Console.WriteLine(myReader.GetString(0))
    End While
    myReader.Close()
    myConnection.Close()
 End Sub

[C#]
public void CreateMyIfxDataReader(string mySelectQuery,string myConnectionString)
{
   IfxConnection myConnection = new IfxConnection(myConnectionString);
   IfxCommand myCommand = new IfxCommand(mySelectQuery, myConnection);
   myCommand.Connection.Open();
   IfxDataReader myReader =
   myCommand.ExecuteReader(CommandBehavior.CloseConnection);
   while(myReader.Read())
   {
      Console.WriteLine(myReader.GetString(0));
   }
   myReader.Close();
   myConnection.Close();
}
```

#### Reference

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

*IfxCommand.ExecuteReader () Method:*

Sends the CommandText to the Connection and builds a IfxDataReader.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Overloads Public Function ExecuteReader() As IfxDataReader
[C#]
public IfxDataReader
 ExecuteReader();
[C++]
public: IfxDataReader
```

```
* ExecuteReader();
[JScript]
public function ExecuteReader() : IfxDataReader;
```

**Return value**

A IfxDataReader object.

**Remarks**

To prepare for the execution of a stored procedure, set the CommandType property to StoredProcedure, and set the CommandText property to the name of the stored procedure. Now, when you call ExecuteReader, the application will execute this stored procedure.

You can concurrently access data from multiple IfxDataReader instances that use the same IfxConnection instance. Each IfxDataReader instance must be associated with its own IfxCommand instance.

**Example**

[Visual Basic, C#] The following example creates a IfxCommand by passing a string with an SQL SELECT statement, and a connection string. The SQL is then executed by calling ExecuteReader.

```
[Visual Basic]
Public Sub CreateMyIfxDataReader(mySelectQuery As String, _
myConnectionString As String)
    Dim myConnection As New IfxConnection(myConnectionString)
    Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
    myCommand.Connection.Open()
    Dim myReader As IfxDataReader = myCommand.ExecuteReader()
    Try
      While myReader.Read()
          Console.WriteLine(myReader.GetString(0))
      End While
    Finally
      myReader.Close()
      myConnection.Close()
    End Try
 End Sub

[C#]
public void CreateMyIfxDataReader(string mySelectQuery,string myConnectionString)
{
   IfxConnection myConnection = new IfxConnection(myConnectionString);
   IfxCommand myCommand = new IfxCommand(mySelectQuery, myConnection);
   myCommand.Connection.Open();
   IfxDataReader myReader = myCommand.ExecuteReader();
   try
   {
     while(myReader.Read())
     {
        Console.WriteLine(myReader.GetString(0));
     }
   }
   finally
   {
     myReader.Close();
     myConnection.Close();
   }
}
```

**Reference**

"IfxCommand Class" on page 5-110
"IfxCommand Members" on page 5-112
"IBM.Data.Informix Namespace" on page 5-1
"IfxCommand.ExecuteReader Method" on page 5-129

*IfxCommand.ExecuteReader (CommandBehavior) Method:*

Sends the CommandText to the Connection, and builds a IfxDataReader using one of the CommandBehavior values.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function ExecuteReader( _
   ByVal behavior As CommandBehavior _
) As IfxDataReader

[C#]
public IfxDataReader
 ExecuteReader(
   CommandBehavior behavior
);
[C++]
public: IfxDataReader
* ExecuteReader(
   CommandBehavior behavior
);
[JScript]
public function ExecuteReader(
   behavior : CommandBehavior
) : IfxDataReader
;
```

**Parameters**

**behavior**
One of the System.Data.CommandBehavior values.

**Return value**

A IfxDataReader instance.

**Remarks**

If you expect your SQL statement to return only a single row, specifying SingleRow as the CommandBehavior value may improve application performance.

The IfxDataReader supports a special mode that enables large binary values to be read efficiently. For more information, see the SequentialAccess setting for CommandBehavior.

You can concurrently access data from multiple IfxDataReader instances that use the same IfxConnection instance. Each IfxDataReader instance must be associated with its own IfxCommand instance.

### Example

[Visual Basic, C#] The following example creates a IfxCommand , then executes it by passing a string that is an SQL SELECT statement, and a string to use to connect to the database. CommandBehavior is then set to CloseConnection.

```
[Visual Basic]
Public Sub CreateMyIfxDataReader(mySelectQuery As String, _
myConnectionString As String)
    Dim myConnection As New IfxConnection(myConnectionString)
    Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
    myCommand.Connection.Open()
    Dim myReader As IfxDataReader =
        myCommand.ExecuteReader(CommandBehavior.CloseConnection)
    While myReader.Read()
        Console.WriteLine(myReader.GetString(0))
    End While
    myReader.Close()
    myConnection.Close()
 End Sub

[C#]
public void CreateMyIfxDataReader(string mySelectQuery,string myConnectionString)
{
   IfxConnection myConnection = new IfxConnection(myConnectionString);
   IfxCommand myCommand = new IfxCommand(mySelectQuery, myConnection);
   myCommand.Connection.Open();
   IfxDataReader myReader =
   myCommand.ExecuteReader(CommandBehavior.CloseConnection);
   while(myReader.Read())
   {
      Console.WriteLine(myReader.GetString(0));
   }
   myReader.Close();
   myConnection.Close();
}
```

#### Reference

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

"IfxCommand.ExecuteReader Method" on page 5-129

## IfxCommand.ExecuteRow Method:

Sends CommandText to the Connection and builds a IfxRecord. This method is intended for result sets with a single row.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function ExecuteRow () As IfxRecord
[C#]
public IfxRecord ExecuteRow()
[C++]
public:
IfxRecord ExecuteRow ()
[JScript]
public function ExecuteRow () : IfxRecord
```

### Return value

A IfxRecord instance representing the first (or only) row of a result set.

### Example

[C#] The following example demonstrates how to create a IfxRecord and read its column data.

```
[C#]
  public static string getSalesData(IfxConnection conn)
  {
    string salesQuery = "SELECT MAX(SALES) FROM SALES";
    string salesData = "";
    IfxCommand cmd = new IfxCommand(salesQuery, conn);
    IfxRecord salesRec = cmd.ExecuteRow();

    salesData = salesRec.GetIfxInt32(0).ToString();

    return salesData;
  }
```

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxCommand Class" on page 5-110

"IBM.Data.Informix Namespace" on page 5-1

## IfxCommand.ExecuteScalar Method:

Executes the query, and returns the first column of the first row in the resultset returned by the query. Extra columns or rows are ignored.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
NotOverridable Public Function ExecuteScalar() As Object
[C#]
public object ExecuteScalar();
[C++]
public: __sealed Object* ExecuteScalar();
[JScript]
public function ExecuteScalar() : Object;
```

### Return value

The first column of the first row in the resultset.

### Remarks

Use the ExecuteScalar method to retrieve a single value (for example, an aggregate value) from a database. This requires less code than using the ExecuteReader

method, and then performing the operations necessary to generate the single value from the data returned by a IfxDataReader .

A typical ExecuteScalar query can be formatted as in the following C# example:

```
CommandText = "select count(*) as NumberOfEmployee from EMPLOYEE";
Int count = (int) ExecuteScalar();
```

**Example**

[Visual Basic, C#] The following example creates a IfxCommand and then executes it using ExecuteScalar. The example is passed a string that is an SQL statement that returns an aggregate result, and a string to use to connect to the database.

```
[Visual Basic]
Public Sub CreateMyIfxCommand(myScalarQuery As String,
    myConnection As IfxConnection)
    Dim myCommand As New IfxCommand(myScalarQuery, myConnection)
    myCommand.Connection.Open()
    Dim qryValue As object = myCommand.ExecuteScalar()
    myConnection.Close()
End Sub 'CreateMyIfxCommand

[C#]
public void CreateMyIfxCommand(string myScalarQuery, IfxConnection myConnection)
 {
    IfxCommand myCommand = new IfxCommand(myScalarQuery, myConnection);
    myCommand.Connection.Open();
    object qryValue = myCommand.ExecuteScalar();
    myConnection.Close();
 }
```

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.Prepare Method:**

Creates a prepared (or compiled) version of the command at the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Public Sub Prepare()
[C#]
public void Prepare();
[C++]
public: __sealed void Prepare();
[JScript]
public function Prepare();
```

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidOperationException | The Connection is not set.<br><br>-or-<br><br>The Connection is not IfxConnection.Open. |

**Remarks**

Before you call Prepare, specify the data type of each parameter in the statement to be prepared. For each parameter that has a variable length data type, you must set the "IfxParameter.Size property to the maximum size needed. Prepare returns an error if these conditions are not met.

If you call an Execute method after calling Prepare, any parameter value that is larger than the value specified by the Size property is automatically truncated to the original specified size of the parameter, and no truncation errors are returned.

Output parameters (whether prepared or not) must have a user-specified data type. If you specify a variable length data type, you must also specify the maximum Size.

**Example**

[Visual Basic, C#] The following example creates a database connection by passing a string to connect to the database, creates a IfxCommand by passing in a string that is an SQL SELECT statement and a connection object and opens the connection. The example then prepares the command (the SQL SELECT statement passed in previously) on the database.

```
[Visual Basic]
Public Sub CreateMyIfxCommand(mySelectQuery As String, _
myConnectionString As String)
    Dim myConnection As New IfxConnection(myConnectionString)
    Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
    myCommand.Connection.Open()
    myCommand.Prepare()
End Sub

[C#]
public void CreateMyIfxCommand(string mySelectQuery, string myConnectionString)
{
   IfxConnection myConnection = new IfxConnection(myConnectionString);
   IfxCommand myCommand = new IfxCommand(mySelectQuery, myConnection);
   myCommand.Connection.Open();
   myCommand.Prepare();
}
```

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.ResetCommandTimeout Method:**

Resets the CommandTimeout property to the default value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub ResetCommandTimeout()
[C#]
public void ResetCommandTimeout();
[C++]
public: void ResetCommandTimeout();
[JScript]
public function ResetCommandTimeout();
```

**Remarks**

The default value of the CommandTimeout is 30 seconds.

**Example**

[Visual Basic, C#] The following example creates a IfxCommand, sets the CommandTimeout, displays the property, resets the CommandTimeout, and displays the property again. To accomplish this, the method is passed a string that is an SQL SELECT statement and a string to use to connect to the database.

```
[Visual Basic]
Public Sub CreateMyIfxCommand(mySelectQuery As String, _
myConnectionString As String)
    Dim myConnection As New IfxConnection(myConnectionString)
    Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
    myCommand.CommandTimeout = 15
    MessageBox.Show(myCommand.CommandTimeout.ToString())
    myCommand.ResetCommandTimeout()
    MessageBox.Show(myCommand.CommandTimeout.ToString())
End Sub

[C#]
public void CreateMyIfxCommand(string mySelectQuery, string myConnectionString)
{
   IfxConnection myConnection = new IfxConnection(myConnectionString);
   IfxCommand myCommand = new IfxCommand(mySelectQuery, myConnection);
   myCommand.CommandTimeout = 15;
   MessageBox.Show(myCommand.CommandTimeout.ToString());
   myCommand.ResetCommandTimeout();
   MessageBox.Show(myCommand.CommandTimeout.ToString());
}
```

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

## IfxCommand Properties

The properties of the IfxCommand class are listed here. For a complete list of IfxCommand class members, see the IfxCommand Members topic.

## Public Properties

| Property | Description |
|---|---|
| CommandText | Gets or sets the SQL statement, or stored procedure to run against the database. |
| CommandTimeout | Gets or sets the wait time before terminating the execution of a command and generating an error. |
| CommandType | Gets or sets a value indicating how the CommandText property is interpreted. |
| Connection | Gets or sets the IfxConnection that is used by an instance of the IfxCommand. |
| Container (inherited from Component) | Gets the IContainer that contains the Component. |
| IfxTypeOutput | Determines whether the output-only parameter values associated with the IfxCommand are returned as native DB2 data types (specifically, classes and structures in the IBM.Data.IfxTypes namespace). |
| DbConnection | Gets or sets the DbConnection used by an instance of the IfxCommand. |
| DbParameterCollection | Gets the DbParameterCollection object. |
| DbTransaction | Gets or sets the DbTransaction within which the IfxCommand runs. |
| DesignTimeVisible | Gets or sets a boolean value indicating whether an IfxCommand generated by a data adapter is visible. |
| DisableCursorHold | Controls the effect of a transaction completion on open cursors. |
| EnableExtendedIndicators | Enables the use of default and unassigned indicators as parameters. |
| FitHighPrecisionType | Gets or sets the type of precision to be used. The acceptable values are as follows:<br>• `WithTruncate`. Results in returning a .NET system type after silently truncating the column value if needed.<br>• `AsString`. Results in converting the column to a .NET string type.<br>• `ReturnException`. Results in a truncation exception if the value does not fit in the .NET system type. |
| Parameters | Gets the IfxParameterCollection . |
| ResultSetAsReturnValue | Set this value to retrieve result sets from Informix user-defined routines (UDR) as a ReturnValue parameter. This property is ignored when a ReturnValue parameter is not bound in the parameter collection. |

| Property | Description |
|---|---|
| Site (inherited from Component) | Gets or sets the ISite of the Component. |
| StatementConcentrator | Gets or sets the value determining whether or not to enable statement concentrator literals. |
| Transaction | Gets or sets the IfxTransaction within which the IfxCommand runs. |
| UpdatedRowSource | Gets or sets a value that specifies how the Update method applies command results to the DataRow. |

## Protected Properties

| Property | Description |
|---|---|
| DesignMode (inherited from Component) | Gets a value that indicates whether the Component is in design mode. |
| Events (inherited from Component) | Gets the list of event handlers that are attached to a Component. |

**Reference**

"IfxCommand Class" on page 5-110

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.CommandText Property:**

Gets or sets the SQL statement, XQuery expression, or stored procedure to execute against the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property CommandText As String
[C#]
public string CommandText {get; set;}
[C++]
public: __property String* get_CommandText();
public: __property void set_CommandText(String*);
[JScript]
public function get CommandText() : String;
public function set CommandText(String);
```

**Property value**

The SQL statement, or stored procedure to execute. The default value is an empty string ("").

**Stored procedure remarks**

When the CommandType property is set to StoredProcedure, the CommandText property should be set to the name of the stored procedure. Using CommandType.StoredProcedure is the recommended method of invoking a stored procedure.

A procedure is an executable object stored at the database. Generally, it is one or more SQL statements that have been precompiled. The syntax for calling a procedure with CommandType.Text is

```
{[?=]call procedure-name[([parameter][,[parameter]]...)]}
```

where **procedure-name** specifies the name of a procedure and **parameter** specifies a procedure parameter. The escape sequence (the curly brackets) must be specified when "?=" is used for the return parameter. Otherwise, the escape sequence is optional.

The command executes this stored procedure when you call one of the Execute methods (for example, ExecuteReader or ExecuteNonQuery).

**SQL statement remarks**

You cannot set the Connection, CommandType and CommandText properties if the current connection is performing an execute or fetch operation.

When passing parameters to SQL statements or stored procedures called by a IfxCommand, the IBM Data Server Provider for .NET supports named parameters or positioned parameters using parameter markers. You cannot use a combination of named parameters or positioned parameters in the same SQL statement.

When using named parameters in a IfxParameterCollection, specify the name of the parameter object in the SQL statement, and add the parameter object to the DB2Command object. For example:

```
IfxCommand cmd = new IfxCommand( "SELECT * FROM EMPLOYEE
 WHERE LASTNAME = @lastname AND WORKDEPT = @workdept", conn );
cmd.Parameters.Clear();
cmd.Parameters.Add( "@workdept", IfxType.SmallInt );
cmd.Parameters.Add( "@lastname", IfxType.VarChar, 15 );
IfxReader reader = cmd.ExecuteReader();
```

Parameter marker names are case-insensitive, must be prefixed by the symbol '@'or by a colon ':' for host variables, and can be made up of any symbol that can be used as part of an SQL identifier. For details regarding SQL identifiers, see the topic: “Identifiers” in the DB2 data server documentation. Using more than one type of parameter within the same statement is not supported. This means that a statement using positioned parameter markers cannot contain named parameter markers or host variables. Likewise, a statement using named parameter markers cannot contain host variables or positioned parameter markers

Support for host variables, prefixed by a colon ':', is disabled by default. To enable host variable support the HostVarParameters property must be set to TRUE in the connection string. Host variable support has been added for compatibility with applications that already contain them. Developing new applications with host variables is not recommended.

```
SELECT * FROM Customers WHERE CustomerID = ?
```

As a result, the order in which IfxParameter objects are added to the IfxParameterCollection must directly correspond to the position of the question mark placeholder for the parameter.

If a parameter contains a null value, the IBM Data Server Provider for .NET will bind that parameter as a null value. For example, the IfxParameterCollection:

```
{1, null, 2}
```

passed into the CommandText property:

```
 call sp(?, ?, ?)
```

results in the IBM Data Server Provider for .NET binding the first parameter to the value 1, the second parameter to the null value, and the third parameter to the value 2.

**Example**

[Visual Basic, C#] The following example creates a IfxCommand and sets some of its properties.

```
[Visual Basic]
Public Sub CreateMyIfxCommand()
    Dim myCommand As New IfxCommand()
    myCommand.CommandText = "SELECT * FROM EMPLOYEE ORDER BY EMPNO"
    myCommand.CommandTimeout = 20
End Sub

[C#]
public void CreateMyIfxCommand()
{
   IfxCommand myCommand = new IfxCommand();
   myCommand.CommandText = "SELECT * FROM EMPLOYEE ORDER BY EMPNO";
   myCommand.CommandTimeout = 20;
}
```

**Reference**


"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

"IfxCommand.Connection Property" on page 5-144

"IfxCommand.CommandTimeout Property"

"IfxCommand.CommandType Property" on page 5-142


**IfxCommand.CommandTimeout Property:**

Gets or sets the wait time before terminating the attempt to execute a command, or the execution of a command. An error is generated after termination.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property CommandTimeout As Integer
[C#]
public int CommandTimeout {get; set;}
```

```
[C++]
public: __property int get_CommandTimeout();
public: __property void set_CommandTimeout(int);
[JScript]
public function get CommandTimeout() : int;
public function set CommandTimeout(int);
```

**Property value**

The time (in seconds) to wait for the command to execute. The default is 30 seconds.

**Exceptions**

| Exception type | Condition |
|---|---|
| ArgumentException | The property value assigned is less than 0. |

**Remarks**

A value of zero (0) specifies no limit to the wait time, rather than no wait time, and therefore should be avoided.

**Example**

[Visual Basic, C#] The following example creates a IfxCommand and sets some of its properties.

```
[Visual Basic]
Public Sub CreateMyIfxCommand()
    Dim mySelectQuery As String = "SELECT * FROM EMPLOYEE ORDER BY EMPNO"
    Dim myCommand As New IfxCommand(mySelectQuery)
    myCommand.CommandTimeout = 20
End Sub

[C#]
public void CreateMyIfxCommand()
{
   string mySelectQuery = "SELECT * FROM EMPLOYEE ORDER BY EMPNO";
   IfxCommand myCommand = new IfxCommand(mySelectQuery);
   myCommand.CommandTimeout = 20;
}
```

**Reference**


"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

"IfxCommand.Connection Property" on page 5-144

"IfxCommand.CommandText Property" on page 5-139

"IfxCommand.CommandType Property"

"IfxCommand.CommandTimeout Property" on page 5-141


**IfxCommand.CommandType Property:**

Gets or sets a value indicating how the CommandText property is interpreted.

**Namespace:**

IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property CommandType As CommandType
[C#]
public CommandType CommandType {get; set;}
[C++]
public: __property CommandType get_CommandType();
public: __property void set_CommandType(CommandType);
[JScript]
public function get CommandType() : CommandType;
public function set CommandType(CommandType);
```

**Property value**

One of the System.Data.CommandType values. The default is Text.

**Exceptions**

| Exception type | Condition |
|---|---|
| ArgumentException | The value was not a valid CommandType. |

**Remarks**

When the CommandType property is set to StoredProcedure, you should set the CommandText property to the name of the stored procedure. The command then executes this stored procedure when you call one of the Execute methods (for example, ExecuteReader or ExecuteNonQuery).

The Connection, CommandType and CommandText properties cannot be set if the current connection is performing an execute or fetch operation.

When passing parameters to SQL statements or stored procedures called by a IfxCommand, the IBM Data Server Provider for .NET supports named parameters or positioned parameters using parameter markers.

For more information, see "Using Stored Procedures with a Command" in the Microsoft(R) .NET Framework SDK documentation.

**Example**

[Visual Basic, C#] The following example creates an instance of a derived class IfxCommand and sets some of its properties.

```
[Visual Basic]
Public Sub CreateMyIfxCommand()
    Dim myCommand As New IfxCommand()
    myCommand.CommandText = "SELECT * FROM EMPLOYEE ORDER BY EMPNO"
    myCommand.CommandType = CommandType.Text
End Sub

[C#]
public void CreateMyIfxCommand()
{
```

```
   IfxCommand myCommand = new IfxCommand();
   myCommand.CommandText = "SELECT * FROM EMPLOYEE ORDER BY EMPNO";
   myCommand.CommandType = CommandType.Text;
}
```

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

"IfxCommand.CommandText Property" on page 5-139

"IfxCommand.UpdatedRowSource Property" on page 5-156

### IfxCommand.Connection Property:

Gets or sets the IfxConnection used by this instance of the IfxCommand.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Connection As IfxConnection

[C#]
public IfxConnection
 Connection {get; set;}
[C++]
public: __property IfxConnection
* get_Connection();
public: __property void set_Connection(IfxConnection
*);
[JScript]
public function get Connection() : IfxConnection
;
public function set Connection(IfxConnection
);
```

**Property value**

The connection to a database. The default is a null value.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidOperationException | The Connection property was changed while a transaction was in progress. |

**Remarks**

You cannot set the Connection, CommandType, and CommandText properties if the current connection is performing an execute or fetch operation.

If you set Connection while a transaction is in progress and the Transaction property is not null, an InvalidOperationException is generated. If you set Connection after the transaction has been committed or rolled back, and the

Transaction property is not null, the Transaction property is then set to a null value.

**Example**

[Visual Basic, C#] The following example creates a IfxCommand and sets some of its properties.

```
[Visual Basic]
Public Sub CreateMyIfxCommand()
    Dim mySelectQuery As String = _
       "SELECT * FROM EMPLOYEE ORDER BY EMPNO"
    Dim myCommand As New IfxCommand(mySelectQuery)
    myCommand.Connection = New IfxConnection _
       ("DATABASE=SAMPLE;")
End Sub

[C#]
public void CreateMyIfxCommand()
{
   string mySelectQuery = "SELECT * FROM EMPLOYEE ORDER BY EMPNO";
   IfxCommand myCommand = new IfxCommand(mySelectQuery);
   myCommand.Connection = new IfxConnection
      ("DATABASE=SAMPLE;");
}
```

**Reference**


"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

"IfxCommand.CommandText Property" on page 5-139

"IfxCommand.CommandTimeout Property" on page 5-141

"IfxCommand.CommandType Property" on page 5-142


**IfxCommand.IfxTypeOutput Property:**

Determines if the output-only parameter values associated with the IfxCommand will be returned as native Informix data types (specifically, classes and structures in the IBM.Data.IfxTypes namespace).

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
Public Overrides Property IfxTypeOutput As Boolean
[C#]
public override bool IfxTypeOutput { get; set; }
[C++]
public:
virtual property bool IfxTypeOutput {
   bool get () override;
   void set (bool value) override;
}
```

**Property value**

A boolean value that determines if the associated output-only IfxParameter instances will return column values as native Informix data type instances (for example, IfxString).

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.DbConnection Property:**

Gets or sets the DbConnection used by this instance of the IfxCommand.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property DbConnection As DbConnection
[C#]
public DbConnection DbConnection {get; set;}
[C++]
public: __property DbConnection* get_DbConnection();
public: __property void set_DbConnection(DbConnection*);
[JScript]
public function get DbConnection() : DbConnection;
public function set DbConnection(DbConnection);
```

**Property value**

The connection to the data source.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommand Class" on page 5-110

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.DbParameterCollection Property:**

Gets the DbParameterCollection object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property DbParameterCollection As DbParameterCollection
[C#]
public DbParameterCollection DbParameterCollection {get;}
[C++]
public: __property DbParameterCollection* get_DbParameterCollection();
[JScript]
public function get DbParameterCollection() : DbParameterCollection;
```

**Property value**

The parameters of the SQL statement or stored procedure.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommand Class" on page 5-110

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.DbTransaction Property:**

Gets or sets the DbTransaction within which the IfxCommand executes.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property DbTransaction As DbTransaction
[C#]
public DbTransaction DbTransaction {get; set;}
[C++]
public: __property DbTransaction* get_DbTransaction();
public: __property void set_DbTransaction(DbTransaction*);
[JScript]
public function get DbTransaction() : DbTransaction;
public function set DbTransaction(DbTransaction);
```

**Property value**

The transaction within which a IfxCommand object executes.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommand Class" on page 5-110

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.DesignTimeVisible Property:**

Gets or sets a boolean value indicating whether or not a IfxCommand generated by a data adapter is visible.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Property DesignTimeVisible As Boolean
[C#]
public override bool DesignTimeVisible { get; set; }
[C++]
public:
virtual property bool DesignTimeVisible {
   bool get () override;
   void set (bool value) override;
}
[JScript]
public override function get DesignTimeVisible () : boolean
public override function set DesignTimeVisible (value : boolean)
```

**Property value**

Gets or sets a boolean value indicating whether or not a IfxCommand generated by a data adapter is visible.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

**DB2Command.DisableCursorHold Property:**

Specifies whether the cursors that a .NET application opens on the server should be left open after committing a transaction.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property DisableCursorHold As Boolean
[C#]
public bool DisableCursorHold {set;}
[C++]
public: __property bool set_DisableCursorHold();
[JScript]
public function set DisableCursorHold():Boolean
```

**Property value**

If you set the DisableCursorHold property to `true`, the cursors that the .NET application opens on the server are closed before the application commits the transaction. Setting DisableCursorHold property to `true` reduces the cost for applications that do not intend to use the cursor after the transaction is committed.

The default value of the property is `false`.

**Remarks**

For XA connections, DisableCursorHold is set to `true` and cannot be set to `false`.

If you specify an invalid value for the DisableCursorHold property, an ArgumentException is thrown.

**Version information**

**.NET Framework version**
Supported in 2.0, 3.0, 3.5, and 4.0

**IfxCommand.EnableExtendedIndicators Property:**

Enables the use of default and unassigned indicators as parameters.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
Public Overrides Property EnableExtendedIndicators As Boolean
[C#]
public override bool EnableExtendedIndicators { get; set; }
[C++]
public:
virtual property bool EnableExtendedIndicators {
   bool get () override;
   void set (bool value) override;
}
```

**Property value**

A boolean value that determines whether extended indicators should be enabled. If the value is `true`, extended indicators are enabled; if the value is `false`, extended indicators are not enabled. The default value is `false`.

**Remarks**

Enabling extended indicators requires extra processing. Enable them only if the parameters that you are passing are indicated as a default or unspecified value.

Assume that you have the following table definition:

```
CREATE TABLE T1 ( C1 INT WITH DEFAULT 100, C2 INT, C3 VARCHAR(6) )
```

The following example shows a named parameter that uses a default indicator:

```
[Visual Basic]

[C#]
DB2Connection myConn = new DB2Connection(myConnString);
DB2Command myCmd = conn.CreateCommand();
myCmd.EnableExtendedIndicators = true;
myCmd.CommandText = "INSERT INTO T1 VALUES(@p1, @p2, @p3)";
myCmd.Parameters.Add( new DB2Parameter("p1", DB2Parameter.Default) );
myCmd.Parameters.Add( new DB2Parameter("p2", 123) );
myCmd.Parameters.Add( new DB2Parameter("p3", "abcd") );
```

The following example shows a positioned parameter that uses an unassigned indicator:

```
[Visual Basic]

[C#]
DB2Connection conn = new DB2Connection(connString);
DB2Command cmd = conn.CreateCommand();
cmd.EnableExtendedIndicators = true;
cmd.CommandText = "UPDATE T1 SET C1=? C2=? where C3=?";
cmd.Parameters.Add(DB2Parameter.Unassigned);
cmd.Parameters.Add( new DB2Parameter(null, 123) );
cmd.Parameters.Add( new DB2Parameter(null, "abcd") );
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**IfxCommand.Parameters Property:**

Gets the IfxParameterCollection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Parameters As IfxParameterCollection
[C#]
public IfxParameterCollection Parameters {get;}
[C++]
public: __property IfxParameterCollection * get_Parameters();
[JScript]
public function get Parameters() : IfxParameterCollection;
```

**Property value**

The parameters of the SQL statement or stored procedure. The default is an empty collection.

**Remarks**

When passing parameters to SQL statements or stored procedures called by a IfxCommand, the IBM Data Server Provider for .NET supports host variables, named parameters, and positioned parameters using parameter markers. You cannot use a combination of host variables, named parameters or positioned parameters in the same SQL statement.

**Host variables**

When using host variables in a IfxParameterCollection, specify the name of the parameter object in the SQL statement. For example:

```
SELECT * FROM EMPLOYEE
  WHERE FIRSTNME = :firstname
  AND LASTNAME = :lastname
  AND WORKDEPT = :workdept
```

Parameter names are case-insensitive, must be prefixed by the symbol ':', and can be made up of any symbol that can be used as part of an SQL identifier.

Support for host variables, prefixed by a colon ':', is disabled by default. To enable host variable support the HostVarParameters property must be set to TRUE in the connection string.

**Named parameters**

When using named parameters in a IfxParameterCollection, specify the name of the parameter object in the SQL statement. For example:

```
SELECT * FROM EMPLOYEE
  WHERE FIRSTNME = @firstname
  AND LASTNAME = @lastname
  AND WORKDEPT = @workdept
```

Parameter names are case-insensitive, must be prefixed by the symbol '@', and can be made up of any symbol that can be used as part of an SQL identifier.

**Positioned Parameters**

When using positioned parameters, use the question mark (?) parameter marker. For example:

```
SELECT * FROM EMPLOYEE WHERE EMPNO = ?
```

The order in which IfxParameter objects are added to the IfxParameterCollection must directly correspond to the position of the question mark placeholder for the parameter in the command text.

**Note:** If the parameters in the collection do not match the requirements of the query to be executed, an error may result.

**Example**

[Visual Basic, C#] The following examples create a IfxCommand and displays its parameters. To accomplish this, the method is passed a IfxConnection , a query string that is an SQL SELECT statement, and an array of IfxParameter objects. For these examples, this array contains a single IfxParameter object, with the name "@param1".

```
[Visual Basic]
'Using host variables
Public Sub CreateMyIfxCommand(myConnection As IfxConnection, _
 mySelectQuery As String, myParamArray() As IfxParameter)
     Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
     myCommand.CommandText = "SELECT ID, NAME FROM STAFF WHERE ID = :param1"
     myCommand.Parameters.Add(myParamArray)
     Dim j As Integer
     For j = 0 To myCommand.Parameters.Count - 1
        myCommand.Parameters.Add(myParamArray(j))
     Next j
     Dim myMessage As String = ""
     Dim i As Integer
```

```
        For i = 0 To myCommand.Parameters.Count - 1
            myMessage += myCommand.Parameters(i).ToString() + ControlChars.Cr
        Next i
        Console.WriteLine(myMessage)
    End Sub

    'Using named parameters
    Public Sub CreateMyIfxCommand(myConnection As IfxConnection, _
    mySelectQuery As String, myParamArray() As IfxParameter)
        Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
        myCommand.CommandText = "SELECT ID, NAME FROM STAFF WHERE ID = @param1"
        myCommand.Parameters.Add(myParamArray)
        Dim j As Integer
        For j = 0 To myCommand.Parameters.Count - 1
           myCommand.Parameters.Add(myParamArray(j))
        Next j
        Dim myMessage As String = ""
        Dim i As Integer
        For i = 0 To myCommand.Parameters.Count - 1
            myMessage += myCommand.Parameters(i).ToString() + ControlChars.Cr
        Next i
        Console.WriteLine(myMessage)
    End Sub

    'Using positioned parameters
    Public Sub CreateMyIfxCommand(myConnection As IfxConnection, _
    mySelectQuery As String, myParamArray() As IfxParameter)
        Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
        myCommand.CommandText = "SELECT ID, NAME FROM STAFF WHERE ID = ?"
        myCommand.Parameters.Add(myParamArray)
        Dim j As Integer
        For j = 0 To myCommand.Parameters.Count - 1
           myCommand.Parameters.Add(myParamArray(j))
        Next j
        Dim myMessage As String = ""
        Dim i As Integer
        For i = 0 To myCommand.Parameters.Count - 1
            myMessage += myCommand.Parameters(i).ToString() + ControlChars.Cr
        Next i
        Console.WriteLine(myMessage)
    End Sub


    [C#]
    //Using host variables
    public void CreateMyIfxCommand(IfxConnection myConnection,
    string mySelectQuery, IfxParameter[] myParamArray) {
       // assume myConnection has HostVarParameters set to TRUE
       IfxCommand myCommand = new IfxCommand(mySelectQuery, myConnection);
       myCommand.CommandText = "SELECT ID, NAME FROM STAFF WHERE ID = :param1";
       myCommand.Parameters.Add(myParamArray);
       for (int j=0; j<myParamArray.Length; j++)
       {
          myCommand.Parameters.Add(myParamArray[j]) ;
       }
       string myMessage = "";
       for (int i = 0; i < myCommand.Parameters.Count; i++)
       {
          myMessage += myCommand.Parameters[i].ToString() + "\n";
       }
       MessageBox.Show(myMessage);
    }

    //Using named parameters
    public void CreateMyIfxCommand(IfxConnection myConnection,
    string mySelectQuery, IfxParameter[] myParamArray) {
       IfxCommand myCommand = new IfxCommand(mySelectQuery, myConnection);
```

```
    myCommand.CommandText = "SELECT ID, NAME FROM STAFF WHERE ID = @param1";
    myCommand.Parameters.Add(myParamArray);
    for (int j=0; j<myParamArray.Length; j++)
    {
       myCommand.Parameters.Add(myParamArray[j]) ;
    }
    string myMessage = "";
    for (int i = 0; i < myCommand.Parameters.Count; i++)
    {
       myMessage += myCommand.Parameters[i].ToString() + "\n";
    }
    MessageBox.Show(myMessage);
 }

//Using positioned parameters
public void CreateMyIfxCommand(IfxConnection myConnection,
 string mySelectQuery, IfxParameter[] myParamArray) {
    IfxCommand myCommand = new IfxCommand(mySelectQuery, myConnection);
    myCommand.CommandText = "SELECT ID, NAME FROM STAFF WHERE ID = ?";
    myCommand.Parameters.Add(myParamArray);
    for (int j=0; j<myParamArray.Length; j++)
    {
       myCommand.Parameters.Add(myParamArray[j]) ;
    }
    string myMessage = "";
    for (int i = 0; i < myCommand.Parameters.Count; i++)
    {
       myMessage += myCommand.Parameters[i].ToString() + "\n";
    }
    MessageBox.Show(myMessage);
 }
```

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

"IfxParameter Class" on page 5-457

**IfxCommand.ResultSetAsReturnValue Property:**

Set this value to retrieve result sets from Informix user-defined routines (UDR) as a ReturnValue parameter. This property is ignored when a ReturnValue parameter is not bound in the parameter collection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ReturnSetAsReturnValue As Boolean
[C#]
public bool ReturnSetAsReturnValue {get; set;}
[C++]
public: __property bool get_ReturnSetAsReturnValue();
public: __property void set_ReturnSetAsReturnValue(bool);
[JScript]
public function get ReturnSetAsReturnValue() : Boolean;
public function set ReturnSetAsReturnValue(Boolean);
```

### Property value

Set to `true` to work with Informix UDRs that return result sets. Otherwise, set to `false`. The default value is `false`.

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

### Remarks

If this property is set, and the UDR returns multiple rows or columns, the ReturnValue parameter contains the value of the first row and first column. Other values are ignored and are inaccessible.

This property does not affect IfxCommandBuilder.DeriveParameters(). Applications should specify the correct type for the ReturnValue parameter. If the type does not match the ReturnValue parameter, an exception is generated.

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

## IfxCommand.StatementConcentrator Property:

Gets or sets the value determining whether or not to enable statement concentrator literals.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Property StatementConcentrator As string
[C#]
public string StatementConcentrator {get; set;}
[C++]
public: __property string* get_StatementConcentrator();
public: __property void set_StatementConcentrator(string*);
[JScript]
public function get StatementConcentrator() : string;
public function set StatementConcentrator(string);
```

### Property value

A string representing wether or not statement concentrator literals are enabled.

Set the property to:

- `Off` - statement concentrator literals are disabled.
- `Literals` - statement concentrator literals are enabled. Literals will be converted to parameters.

**Remarks**

If this property is not set for the command then the default behavior for statement concentrator literals will be determined by the connection string properties or the server-side configuration.

The value returned by this property reflects the StatementConcentrator value set through the connection string keyword or explicitly set for this property. While this StatementConcentrator property has not been explicitly set, the property value can be changed by either setting the connection string property or by explicitly setting this property. If a StatementConcentrator value has not been specified by either this property or the connection string, the property will return `NULL`.

Once this property has been explicitly set it cannot be changed by the connection string property or be reset to `NULL`. This IfxCommand property will override the connection string property.

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0 3.5, and 4.0

**Reference**

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

**IfxCommand.Transaction Property:**

Gets or sets the IfxTransaction within which the IfxCommand executes.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Transaction As IfxTransaction

[C#]
public IfxTransaction
 Transaction {get; set;}
[C++]
public: __property IfxTransaction
* get_Transaction();
public: __property void set_Transaction(IfxTransaction
*);
[JScript]
public function get Transaction() : IfxTransaction
;
public function set Transaction(IfxTransaction
);
```

**Property value**

A IfxTransaction . The default is a null value.

### Remarks

You cannot set the Transaction property if it is already set to a specific value, and the command is in the process of executing. If you set the transaction property to a IfxTransaction object that is not connected to the same IfxConnection as the IfxCommand object, an exception will be thrown the next time you attempt to execute a statement.

#### Reference

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

## IfxCommand.UpdatedRowSource Property:

Gets or sets a value that specifies how the Update method should apply command results to the DataRow.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Property UpdatedRowSource As UpdateRowSource
[C#]
public UpdateRowSource UpdatedRowSource {get; set;}
[C++]
public: __property UpdateRowSource get_UpdatedRowSource();
public: __property void set_UpdatedRowSource(UpdateRowSource);
[JScript]
public function get UpdatedRowSource() : UpdateRowSource;
public function set UpdatedRowSource(UpdateRowSource);
```

### Property value

One of the System.Data.UpdateRowSource values.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentException | The value entered was not one of the UpdateRowSource values. |

### Remarks

The default UpdateRowSource value is Both unless the command is automatically generated (as in the case of the IfxCommandBuilder ), in which case the default is None.

#### Reference

"IfxCommand Class" on page 5-110

"IfxCommand Members" on page 5-112

"IBM.Data.Informix Namespace" on page 5-1

# IfxConnection Class

Represents a unique session with a data source, for example, a network connection to an IBM Informix server.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## .NET Framework 2.0 3.0, 3.5 and 4.0 Inheritance hierarchy

System.Object
  System.MarshalByRefObject
    System.ComponentModel.Component
      System.Data.Common.DbConnection
        IBM.Data.DB2.IfxConnection

## .NET Framework 2.0 3.0, 3.5 and 4.0 Syntax

```
[Visual Basic]
NotInheritable Public Class IfxConnection
   Inherits DbConnection
   Implements ICloneable, IDbConnection
[C#]
public sealed class IfxConnection : Component, ICloneable,
   IDbConnection
[C++]
public __gc __sealed class IfxConnection : public Component,
   ICloneable, IDbConnection
[JScript]
public class IfxConnection extends Component implements
   ICloneable, IDbConnection
```

## Remarks

A IfxConnection object represents a unique connection to the database specfied in the connection string. In the case of a client/server database system, it is equivalent to a network connection to the server.

The IfxConnection object uses native resources. You should always explicitly close any open IfxConnection objects by calling Close or Dispose before the IfxConnection object goes out of scope. Not doing so leaves the freeing of these native resources to garbage collection, which may not free them immediately. This, in turn, may eventually cause the IBM Data Server Provider for .NET or data server to run out of resources or reach a maximum limit, resulting in sporadic failures. For example, you might encounter Maximum Connections-related errors while a number of connections are waiting to be deleted by the garbage collector. Explicitly closing the connections by calling Close or Dispose allows a more efficient use of native resources, enhancing scalability and improving overall application performance.

**Note:** To deploy high-performance applications, you often need to use connection pooling. However, when you use the IBM Data Server Provider for .NET, you do not need to enable connection pooling because the provider pools connections by default.

If one of the Execute methods of the IfxCommand class results in a fatal IfxException , the IfxConnection may close. However, the user can reopen the connection and continue.

An application that creates an instance of the IfxConnection object can require all direct and indirect callers to have adequate permission to the code by setting declarative or imperative security demands. IfxConnection creates security demands by using the IfxPermission object. Users can verify that their code has adequate permissions by using the IfxPermissionAttribute object. Users and administrators can also use the Code Access Security Policy Tool (Caspol.exe) to modify security policy at the machine, user, and enterprise levels. For more information, see "Securing Applications" in the Microsoft .NET Framework SDK documentation.

## Example

[Visual Basic, C#] The following example creates a IfxCommand and an IfxConnection. The IfxConnection is opened and set as the IfxCommand.Connection property. The example then calls IfxCommand.ExecuteNonQuery, and closes the connection. To accomplish this, the ExecuteNonQuery is passed a connection string and a query string that is an SQL INSERT statement.

```
[Visual Basic]
Public Sub InsertRow(myConnectionString As String)
     ' If the connection string is null, use a default.
     If myConnectionString = "" Then
         myConnectionString = "DATABASE=SAMPLE;"
     End If
     Dim myConn As New IfxConnection(myConnectionString)
     Dim myInsertQuery As String = "INSERT INTO STAFF (ID, NAME) Values(...)"
     Dim myIfxCommand As New IfxCommand(myInsertQuery)
     myIfxCommand.Connection = myConn
     myConn.Open()
     myIfxCommand.ExecuteNonQuery()
     myConn.Close()
 End Sub

[C#]
public void InsertRow(string myConnectionString)
 {
    // If the connection string is null, use a default.
    if(myConnectionString == "")
    {
       myConnectionString = "DATABASE=SAMPLE;";
    }
    IfxConnection myConn = new IfxConnection(myConnectionString);
    string myInsertQuery = "INSERT INTO STAFF (ID, NAME) Values(...)";
    IfxCommand myIfxCommand = new IfxCommand(myInsertQuery);
    myIfxCommand.Connection = myConn;
    myConn.Open();
    myIfxCommand.ExecuteNonQuery();
    myConn.Close();
 }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnection Members"

"IBM.Data.Informix Namespace" on page 5-1

"IfxDataAdapter Class" on page 5-253

"IfxCommand Class" on page 5-110

## IfxConnection Members

The following tables list the members exposed by the IfxConnection class.

### Public Constructors

| Name | Description |
|---|---|
| IfxConnection | Overloaded. Initializes a new instance of the IfxConnection class. |

### Public Properties

| Name | Description |
|---|---|
| CacheData | Gets or sets a boolean value indicating whether data caching is being used. |
| Chaining | Gets a boolean value indicating whether chaining is active. |
| ClientAccountingInformation | Gets or sets the string containing the client accounting information. |
| ClientApplicationInformation | Gets or sets the string containing the client application name. |
| ClientUser | Gets or sets the string that contains the client user ID. |
| ClientWorkstation | Gets or sets the string that contains the name of the client workstation. |
| ConnectionString | Gets or sets the string that is used to open a database connection. |
| Container (inherited from Component) | Gets the IContainer that contains the Component. |
| ConnectionTimeout | Gets or sets a value that determines the limit to the amount of time that an application waits for a connection. |
| Database | Gets the name of the current database or the database to be used after a connection is opened. |
| DataSource | Gets the name of the current database or the database to be used after a connection is opened. |
| ResultSetAsReturnValue | Set this value to work with IDS UDRs that return result sets. |

| Name | Description |
|---|---|
| ServerBuildVersion | Gets a string containing the build version of the server to which the client is connected. |
| ServerMajorVersion | Gets a string containing the major version of the server to which the client is connected. |
| ServerMinorVersion | Gets a string containing the minor version of the server to which the client is connected. |
| ServerRevisionVersion | Gets a string containing the revision version of the server to which the client is connected. |
| ServerType | Gets a string containing the type of server to which the client is connected. |
| ServerVersion | Gets a string containing the version of the server to which the client is connected. |
| Site (inherited from Component) | Gets or sets the ISite of the Component. |
| State | Gets the current state of the connection. |

## Public Methods

| Name | Description |
|---|---|
| BeginChain | Marks the beginning of a chain of insert, update, and delete statements to be sent to the database server. |
| BeginTransaction | Overloaded. Begins a transaction at the database. |
| ChangeDatabase | Changes the current database associated with an open IfxConnection. |
| Close | Closes the connection to the database. This is the preferred method of closing any open connection. |
| CreateCommand | Creates and returns a IfxCommand object associated with the IfxConnection. |
| CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
| Dispose (inherited from Component) | Overloaded. Releases the resources used by the Component. |
| DropDTD | Drops the DTD registered with the database. |
| EndChain | Marks the end of a chain of insert, update, and delete statements to be sent to the database server. |
| EnlistDistributedTransaction | Enlists in the specified transaction as a distributed transaction. |

| Name | Description |
| --- | --- |
| EnlistTransaction (inherited from DbConnection) | Overloaded. Enlists in the specified transaction. |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetDTD | Gets the DTD registered with the database. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| GetSchema() | Overloaded. Returns a DataTable with metadata for the data source associated with this IfxConnection instance. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| Open | Opens a connection to a database with the property settings specified by the ConnectionString . |
| RegisterDTD | Registers the DTD with the database. |
| ReleaseObjectPool | Indicates that the IfxConnection object pool can be released when the last underlying connection is released. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

## Public Events

| Name | Description |
| --- | --- |
| Disposed (inherited from Component) | Adds an event handler to listen to the Disposed event on the component. |
| InfoMessage | Occurs when the IBM Data Server Provider for .NET sends a warning or an informational message. |
| StateChange | Occurs when the state of the connection changes. |

### Protected Properties

| Name | Description |
|---|---|
|  DesignMode (inherited from Component) | Gets a value that indicates whether the Component is currently in design mode. |
|  Events (inherited from Component) | Gets the list of event handlers that are attached to this Component. |

### Protected Methods

| Name | Description |
|---|---|
|  Dispose | Overloaded. Overridden. Releases the resources used by the IfxConnection. |
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft Visual C++, finalizers are expressed using destructor syntax. |
|  GetService (inherited from Component) | Returns an object that represents a service provided by the Component or by its Container. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxConnection Class" on page 5-157

"IBM.Data.Informix Namespace" on page 5-1

## IfxConnection Constructor

Initializes a new instance of the IfxConnection class.

### Overload list

| Name | Description |
|---|---|
| New() | Initializes a new instance of the IfxConnection class. |
| New(String) | Initializes a new instance of the IfxConnection class with the specified connection string. |

### Example

[Visual Basic, C#] The following example creates and opens a IfxConnection.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of the IfxConnection constructor. For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub CreateIfxConnection()
    Dim myConnString As String = "DATABASE=SAMPLE;"
    Dim myConnection As New IfxConnection(myConnString)
    myConnection.Open()
End Sub

[C#]
public void CreateIfxConnection()
{
   string myConnString = "DATABASE=SAMPLE;";
   IfxConnection myConnection = new IfxConnection(myConnString);
   myConnection.Open();
}
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection Constructor ():**

Initializes a new instance of the IfxConnection class.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New()
[C#]
public IfxConnection();
[C++]
public: IfxConnection();
[JScript]
public function IfxConnection();
```

**Remarks**

When a new instance of IfxConnection is created, the read/write properties are set to the following initial values unless they are specifically set using their associated keywords in the ConnectionString property.

| Properties | Initial value |
|---|---|
| ConnectionString | empty string ("") |
| Database | empty string ("") |

You can change the value for these properties only by using the ConnectionString property.

**Example**

[Visual Basic, C#] The following example creates and opens a IfxConnection .

```
[Visual Basic]
Public Sub CreateIfxConnection()
    Dim myConnection As New IfxConnection()
```

```
    myConnection.ConnectionString = "DATABASE=SAMPLE;"
    myConnection.Open()
End Sub

[C#]
public void CreateIfxConnection()
{
   IfxConnection myConnection = new IfxConnection();
   myConnection.ConnectionString = "DATABASE=SAMPLE;";
   myConnection.Open();
}
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

"IfxConnection Constructor" on page 5-162

**IfxConnection Constructor (String):**

Initializes a new instance of the IfxConnection class with the specified connection string.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New( _
   ByVal connectionString As String _
)
[C#]
public IfxConnection(
   string connectionString
);
[C++]
public: IfxConnection(
   String* connectionString
);
[JScript]
public function IfxConnection(
   connectionString : String
);
```

**Parameters**

**connectionString**
The connection used to open the database.

**Remarks**

When a new instance of IfxConnection is created, the read/write properties are set to the following initial values unless they are specifically set using their associated keywords in the ConnectionString property.

| Properties | Initial value |
|---|---|
| ConnectionString | **connectionString** |
| Database | empty string ("") |

You can change the value for these properties only by using the ConnectionString property.

**Example**

[Visual Basic, C#] The following example creates and opens a IfxConnection .

```
[Visual Basic]
Public Sub CreateIfxConnection()
    Dim myConnString As String = "DATABASE=SAMPLE;"
    Dim myConnection As New IfxConnection(myConnString)
    myConnection.Open()
End Sub

[C#]
public void CreateIfxConnection()
{
   string myConnString = "DATABASE=SAMPLE;";
   IfxConnection myConnection = new IfxConnection(myConnString);
   myConnection.Open();
}
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

"IfxConnection Constructor" on page 5-162

## IfxConnection Methods

The methods of the IfxConnection class are listed here. For a complete list of IfxConnection class members, see the IfxConnection Members topic.

### Public Methods

| Method | Description |
|---|---|
|  BeginTransaction | Overloaded. Begins a transaction at the database. |
|  ChangeDatabase | Changes the current database associated with an open IfxConnection . |
|  Close | Closes the connection to the database. This is the preferred method of closing any open connection. |
|  CreateCommand | Creates and returns a IfxCommand object associated with the IfxConnection . |
|  CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
|  DropDTD | Drops the DTD registered with the database. |
|  Dispose (inherited from Component) | Overloaded. Releases the resources used by the Component. |
|  EnlistDistributedTransaction | Enlists in the specified transaction as a distributed transaction. |

| Method | Description |
|---|---|
| EnlistTransaction (inherited from DbConnection) | Overloaded. Enlists in the specified transaction. |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetDTD | Gets the DTD registered with the database. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| GetSchema() | Overloaded. Returns a DataTable with metadata for the data source associated with this IfxConnection instance. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| Open | Opens a connection to a database with the property settings specified by the ConnectionString . |
| RegisterDTD | Registers the DTD with the database. |
| ReleaseObjectPool | Indicates that the IfxConnection object pool can be released when the last underlying connection is released. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

## Protected Methods

| Method | Description |
|---|---|
| Dispose | Overloaded. Overridden. Releases the resources used by the IfxConnection . |
| Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
| GetService (inherited from Component) | Returns an object that represents a service provided by the Component or by its Container. |

| Method | Description |
| --- | --- |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxConnection Class" on page 5-157

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.BeginChain Method:**

Marks the beginning of a chain of insert, update, and delete statements to be sent to the database server.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub BeginChain()
[C#]
public void BeginChain();
[C++]
public: void BeginChain();
[JScript]
public function BeginChain();
```

**Exceptions**

| Exception type | Condition |
| --- | --- |
| InvalidOperationException | The connection is not open. |

**Remarks**

If you need to execute a large batch of insert, update, and delete statements, you can improve application performance by chaining your SQL statements, thereby reducing the number of network flows to the database server.

Before calling the BeginChain method in order to activate chaining, the IfxConnection object must be open. After you call the BeginChain method, the insert, update, and delete statements that you execute against the IfxConnection will be queued on the client until the EndChain method is called. After you call the EndChain method, these statements will be sent to the database server and executed.

Whether chaining is active or inactive, you can execute insert, update, and delete statements using the IfxCommandClass.ExecuteNonQuery method. But when chaining is active, all calls to IfxCommand.ExecuteNonQuery will return -1. Since chained statements are all executed together after the EndChain method is called, the number of rows affected for a particular statement is unknown. Chained statements can be executed from either multiple IfxCommand objects or a single

IfxCommand object. All IfxCommand objects that are used for executing a series of chained statements must be created from the same IfxConnection where chaining was activated.

For connections to database servers that support chaining, the Chaining property is set to true when the BeginChain method is called, and is reset back to false when the EndChain method is called. There are some data servers that do not support chaining. If a server does not support chaining, the IBM Data Server Provider for .NET will ignore the BeginChain and EndChain requests, leave the IfxConnection.Chaining property in a false state, and submit all statements to the database server individually. You can determine if chaining is active by checking the Chaining property.

There is no limit to the number of statements that can be chained for a single IfxConnection object. However, after 2,147,483,646 statements have been queued, the IBM Data Server Provider for .NET will internally close the chain, submit the statements, and restart the chain. Additionally, once the communications buffer between the application and the database server (usually 32KB) is filled, the buffer's contents are sent to the server and saved there until EndChain is called. You can adjust the size of this communications buffer with the **rqrioblk** configuration parameter.

For optimal performance, use parameter markers in your insert, update, and delete statements.

**Example**

[Visual Basic, C#] The following example uses chaining to insert 10000 rows into the STAFF table.

```
[Visual Basic]
Dim con As IfxConnection = new IfxConnection("DATABASE=sample;")
Dim cmd As IfxCommand = con.CreateCommand()
con.Open()

' Initialize an insert statement using parameter markers
cmd.CommandText = "INSERT INTO STAFF(ID) VALUES( ? )"

' Add a parameter
Dim p1 As IfxParameter = cmd.Parameters.Add("@ID", IfxType.Integer )

' Start the chain
con.BeginChain()

Try
   ' Loop to add 10000 rows
   Dim I As Int32
   For I = 1 To 10000
      ' Set the parameter value
      p1.Value = I

      ' Execute the command.
      ' Since chaining is active, this statement is now added
      '   to the chain
      cmd.ExecuteNonQuery()
   Next I

   ' Execute the chain
   con.EndChain()
Catch db2Ex As IfxException
   Dim db2Error As IfxError
```

```
   ' Loop through all the errors
   For Each db2Error in db2Ex.Errors
      Console.WriteLine("SQLSTATE =" & db2Error.SQLState )
      Console.WriteLine("NativeErr=" & db2Error.NativeError )
      Console.WriteLine("RowNumber=" & db2Error.RowNumber )
      Console.WriteLine( db2Error.Message )
   Next IfxError
Finally
   ' Explicitly turn chaining off in case it is still on
   If (con.Chaining) Then
      con.EndChain()
   End If
End Try

con.Close()[C#]
DB2Connection con = new IfxConnection("DATABASE=sample;");
DB2Command cmd = con.CreateCommand();
con.Open();

// Initialize an insert statement using parameter markers
cmd.CommandText = "INSERT INTO STAFF(ID) VALUES( ? )";

// Add a parameter
DB2Parameter p1 = cmd.Parameters.Add("@ID", IfxType.Integer );

// Start the chain
con.BeginChain();

try
{
   // Loop to add 10000 rows
   for( Int32 i = 1; i <= 10000; i++ )
   {
      // Set the parameter value
      p1.Value = i;

      // Execute the command.
      // Since chaining is active, this statement is now added
      //   to the chain
      cmd.ExecuteNonQuery();
   }

   // Execute the chain
   con.EndChain();
}
catch( IfxException db2Ex )
{
   // Loop through all the errors
   foreach( IfxError db2Error in db2Ex.Errors )
   {
      Console.WriteLine("SQLSTATE =" + db2Error.SQLState );
      Console.WriteLine("NativeErr=" + db2Error.NativeError );
      Console.WriteLine("RowNumber=" + db2Error.RowNumber );
      Console.WriteLine( db2Error.Message );
   }
}
finally
{
   // Explicitly turn chaining off in case it is still on
   if( con.Chaining )
   {
      con.EndChain();
   }
}

con.Close();
```

**Reference**

"IfxConnection Class" on page 5-157
"IfxConnection Members" on page 5-159
"IBM.Data.Informix Namespace" on page 5-1
"IfxConnection.EndChain Method" on page 5-181

**IfxConnection.BeginTransaction Method:**

Begins a transaction at the database.

**Overload list**

| Name | Description |
|---|---|
| BeginTransaction() | Begins a transaction at the database. |
| BeginTransaction(IsolationLevel) | Begins a transaction at the database with the specified IsolationLevel value. |

**Example**

[Visual Basic, C#] The following example creates a IfxConnection and a IfxTransaction. It also demonstrates how to use the BeginTransaction, IfxTransaction.Commit, and IfxTransaction.Rollback methods.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of BeginTransaction. For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub RunIfxTransaction(myConnString As String)
    Dim myConnection As New IfxConnection(myConnString)
    myConnection.Open()

    Dim myCommand As New IfxCommand()
    Dim myTrans As IfxTransaction

    ' Start a local transaction
    myTrans = myConnection.BeginTransaction(IsolationLevel.ReadCommitted)
    ' Assign transaction object for a pending local transaction
    myCommand.Transaction = myTrans

    Try
        myCommand.CommandText = "Insert into STAFF (ID, NAME) VALUES (...)"
        myCommand.ExecuteNonQuery()
        myCommand.CommandText = "Insert into STAFF (ID, NAME) VALUES (...)"
        myCommand.ExecuteNonQuery()
        myTrans.Commit()
        Console.WriteLine("Both records are written to database.")
    Catch e As Exception
        myTrans.Rollback()
        Console.WriteLine(e.ToString())
        Console.WriteLine("Neither record was written to database.")
    Finally
        myConnection.Close()
    End Try
End Sub

[C#]
public void RunIfxTransaction(string myConnString)
{
   IfxConnection myConnection = new IfxConnection(myConnString);
   myConnection.Open();
```

```
   IfxCommand myCommand = new IfxCommand();
   IfxTransaction myTrans;

   // Start a local transaction
   myTrans = myConnection.BeginTransaction(IsolationLevel.ReadCommitted);
   // Assign transaction object for a pending local transaction
   myCommand.Transaction = myTrans;

   try
   {
     myCommand.CommandText = "Insert into STAFF (ID, NAME) VALUES (...)";
     myCommand.ExecuteNonQuery();
     myCommand.CommandText = "Insert into STAFF (ID, NAME) VALUES (...)";
     myCommand.ExecuteNonQuery();
     myTrans.Commit();
     Console.WriteLine("Both records are written to database.");
   }
   catch(Exception e)
   {
     myTrans.Rollback();
     Console.WriteLine(e.ToString());
     Console.WriteLine("Neither record was written to database.");
   }
   finally
   {
     myConnection.Close();
   }
}
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

*IfxConnection.BeginTransaction () Method:*

Begins a transaction at the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overloads Public Function BeginTransaction() As IfxTransaction

[C#]
public IfxTransaction
 BeginTransaction();
[C++]
public: IfxTransaction
* BeginTransaction();
[JScript]
public function BeginTransaction() : IfxTransaction
;
```

**Return value**

An object representing the new transaction.

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidOperationException | A transaction is currently active. Parallel transactions are not supported. |

### Remarks

To commit or roll back the transaction, you must explicitly use the IfxTransaction.Commit or IfxTransaction.Rollback methods.

**Note:** If you do not specify an isolation level, the default isolation level for the IBM Data Server Provider for .NET is used. For information about setting the isolation level with the BeginTransaction method, see BeginTransaction Method (IsolationLevel).

### Example

[Visual Basic, C#] The following example creates a IfxConnection and a IfxTransaction . It also demonstrates how to use the BeginTransaction, IfxTransaction.Commit, and IfxTransaction.Rollback methods.

```
[Visual Basic]
Public Sub RunIfxTransaction(myConnString As String)
    Dim myConnection As New IfxConnection(myConnString)
    myConnection.Open()

    Dim myCommand As New IfxCommand()
    Dim myTrans As IfxTransaction

    ' Start a local transaction
    myTrans = myConnection.BeginTransaction()
    ' Assign transaction object for a pending local transaction
    myCommand.Transaction = myTrans

    Try
        myCommand.CommandText = "Insert into STAFF (ID, NAME) VALUES (...)"
        myCommand.ExecuteNonQuery()
        myCommand.CommandText = "Insert into STAFF (ID, NAME) VALUES (...)"
        myCommand.ExecuteNonQuery()
        myTrans.Commit()
        Console.WriteLine("Both records are written to database.")
    Catch e As Exception
        myTrans.Rollback()
        Console.WriteLine(e.ToString())
        Console.WriteLine("Neither record was written to database.")
    Finally
        myConnection.Close()
    End Try
End Sub

[C#]
public void RunIfxTransaction(string myConnString)
{
   IfxConnection myConnection = new IfxConnection(myConnString);
   myConnection.Open();

   IfxCommand myCommand = new IfxCommand();
   IfxTransaction myTrans;

   // Start a local transaction
   myTrans = myConnection.BeginTransaction();
   // Assign transaction object for a pending local transaction
```

```
   myCommand.Transaction = myTrans;

   try
   {
     myCommand.CommandText = "Insert into STAFF (ID, NAME) VALUES (...)";
     myCommand.ExecuteNonQuery();
     myCommand.CommandText = "Insert into STAFF (ID, NAME) VALUES (...)";
     myCommand.ExecuteNonQuery();
     myTrans.Commit();
     Console.WriteLine("Both records are written to database.");
   }
   catch(Exception e)
   {
     myTrans.Rollback();
     Console.WriteLine(e.ToString());
     Console.WriteLine("Neither record was written to database.");
   }
   finally
   {
     myConnection.Close();
   }
}
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

"IfxConnection.BeginTransaction Method" on page 5-170

*IfxConnection.BeginTransaction (IsolationLevel) Method:*

Begins a transaction at the database with the specified IsolationLevel value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overloads Public Function BeginTransaction( _
   ByVal isolevel As IsolationLevel _
) As IfxTransaction

[C#]
public IfxTransaction
 BeginTransaction(
   IsolationLevel isolevel
);
[C++]
public: IfxTransaction
* BeginTransaction(
   IsolationLevel isolevel
);
[JScript]
public function BeginTransaction(
   isolevel : IsolationLevel
) : IfxTransaction
;
```

### Parameters

**isolevel**

The transaction isolation level for this connection. If you do not specify an isolation level, the default isolation level for the IBM Data Server Provider for .NET is used.

### Return value

An object representing the new transaction.

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidOperationException | A transaction is currently active. Parallel transactions are not supported. |

### Remarks

To commit or roll back the transaction, you must explicitly use the IfxTransaction.Commit or IfxTransaction.Rollback methods.

### Example

[Visual Basic, C#] The following example creates a IfxConnection and a IfxTransaction. It also demonstrates how to use the BeginTransaction, IfxTransaction.Commit, and IfxTransaction.Rollback methods.

```
[Visual Basic]
Public Sub RunIfxTransaction(myConnString As String)
    Dim myConnection As New IfxConnection(myConnString)
    myConnection.Open()

    Dim myCommand As New IfxCommand()
    Dim myTrans As IfxTransaction

    ' Start a local transaction
    myTrans = myConnection.BeginTransaction(IsolationLevel.ReadCommitted)
    ' Assign transaction object for a pending local transaction
    myCommand.Transaction = myTrans

    Try
        myCommand.CommandText = "Insert into STAFF (ID, NAME) VALUES (...)"
        myCommand.ExecuteNonQuery()
        myCommand.CommandText = "Insert into STAFF (ID, NAME) VALUES (...)"
        myCommand.ExecuteNonQuery()
        myTrans.Commit()
        Console.WriteLine("Both records are written to database.")
    Catch e As Exception
        myTrans.Rollback()
        Console.WriteLine(e.ToString())
        Console.WriteLine("Neither record was written to database.")
    Finally
        myConnection.Close()
    End Try
End Sub

[C#]
public void RunIfxTransaction(string myConnString)
{
   IfxConnection myConnection = new IfxConnection(myConnString);
   myConnection.Open();
```

```
   IfxCommand myCommand = new IfxCommand();
   IfxTransaction myTrans;

   // Start a local transaction
   myTrans = myConnection.BeginTransaction(IsolationLevel.ReadCommitted);
   // Assign transaction object for a pending local transaction
   myCommand.Transaction = myTrans;

   try
   {
     myCommand.CommandText = "Insert into STAFF (ID, NAME) VALUES (...)";
     myCommand.ExecuteNonQuery();
     myCommand.CommandText = "Insert into STAFF (ID, NAME) VALUES (...)";
     myCommand.ExecuteNonQuery();
     myTrans.Commit();
     Console.WriteLine("Both records are written to database.");
   }
   catch(Exception e)
   {
     myTrans.Rollback();
     Console.WriteLine(e.ToString());
     Console.WriteLine("Neither record was written to database.");
   }
   finally
   {
     myConnection.Close();
   }
}
```

**Reference**


"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

"IfxConnection.BeginTransaction Method" on page 5-170

"IfxTransaction.Commit Method" on page 5-598

"IfxTransaction.Rollback Method" on page 5-601


### IfxConnection.ChangeDatabase Method:

Changes the current database associated with an open IfxConnection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Public Sub ChangeDatabase( _
   ByVal value As String _
) Implements IDbConnection.ChangeDatabase
[C#]
public void ChangeDatabase(
   string value
);
[C++]
public: __sealed void ChangeDatabase(
   String* value
);
```

```
[JScript]
public function ChangeDatabase(
   value : String
);
```

**Implements:**

IDbConnection.ChangeDatabase

**Parameters**

**value** The database name.

**Exceptions**

| Exception type | Condition |
|---|---|
| ArgumentException | The database name is not valid. |
| InvalidOperationException | The connection is not open. |
| IfxException | Cannot change the database. |

**Remarks**

The **value** parameter must contain a valid database name, and cannot contain a null value, an empty string (""), or a string with only blank characters.

**Example**

[Visual Basic, C#] The following example creates a IfxConnection and changes the current database.

```
[Visual Basic]
Public Sub CreateIfxConnection()
     Dim myConnString As String = _
        "DATABASE=SAMPLE"
     Dim myConnection As New IfxConnection(myConnString)
     myConnection.Open()
     myConnection.ChangeDatabase("SAMPLE2")
     MessageBox.Show("Database: " + myConnection.Database.ToString())
     myConnection.Close()[C#]
public void CreateIfxConnection()
 {
    string myConnString = "DATABASE=SAMPLE";
    IfxConnection myConnection = new IfxConnection(myConnString);
    myConnection.Open();
    myConnection.ChangeDatabase("SAMPLE2");
    MessageBox.Show("Database: " + myConnection.Database.ToString());
    myConnection.Close();
 }
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

"IfxConnection.Database Property" on page 5-205

**IfxConnection.Close Method:**

Closes the connection to the database. This is the preferred method of closing any open connection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Public Sub Close() Implements IDbConnection.Close
[C#]
public void Close();
[C++]
public: __sealed void Close();
[JScript]
public function Close();
```

**Implements:**
IDbConnection.Close

**Remarks**

The Close method rolls back any pending transactions. It then releases the connection to the connection pool, or closes the connection if connection pooling is disabled. If Close is called while handling a StateChange event, no additional StateChange events are fired.

An application can call Close more than one time without generating an exception.

**Note:** When you use the IBM Data Server Provider for .NET, you do not need to enable connection pooling because it is on by default.

**Example**

[Visual Basic, C#] The following example creates a IfxConnection , opens it, displays some of its properties, then closes the connection.

```
[Visual Basic]
Public Sub CreateIfxConnection(myConnString As String)
     Dim myConnection As New IfxConnection(myConnString)
     myConnection.Open()
     MessageBox.Show("State: " + myConnection.State.ToString())
     myConnection.Close()
 End Sub

[C#]
public void CreateIfxConnection(string myConnString)
 {
    IfxConnection myConnection = new IfxConnection(myConnString);
    myConnection.Open();
    MessageBox.Show("State: " + myConnection.State.ToString());
    myConnection.Close();
 }
```

In DB2 for z/OS Version 10, if the server has migrated to a different mode during the application execution and if a call to Open has returned a connection with migrated server info, then subsequent calls to Open will always return a connection with migrated server info. All connections established prior to the migration will be recycled on the call to Close.

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1
"IfxConnection.Open Method" on page 5-189

**IfxConnection.CreateCommand Method:**

Creates and returns a IfxCommand object associated with the IfxConnection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function CreateCommand() As IfxCommand

[C#]
public IfxCommand
 CreateCommand();
[C++]
public: IfxCommand
* CreateCommand();
[JScript]
public function CreateCommand() : IfxCommand
;
```

**Return value**

A IfxCommand object.

**Reference**

"IfxConnection Class" on page 5-157
"IfxConnection Members" on page 5-159
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.Dispose Method:**

Releases the resources used by the IfxConnection.

**Overload list**

| Name | Description |
|---|---|
| Dispose(Boolean) | Releases the unmanaged and, optionally, the managed resources used by the IfxConnection. |
| Dispose() | Inherited from Component. |

**Example**

[Visual Basic, C#] The following example creates a IfxConnection and then disposes of it.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of Dispose. For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub IfxConnectionHereAndGone()
    Dim myConnection As New IfxConnection()
    myConnection.Open()
    'Calling Dispose also calls IfxConnection.Close.
    myConnection.Dispose()
End Sub

[C#]
public void IfxConnectionHereAndGone()
{
   IfxConnection myConnection = new IfxConnection();
   myConnection.Open();
   //Calling Dispose also calls IfxConnection.Close.
   myConnection.Dispose();
}
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

*IfxConnection.Dispose (Boolean) Method:*

Releases the unmanaged and, optionally, the managed resources used by the IfxConnection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overrides Overloads Protected Sub Dispose( _
   ByVal disposing As Boolean _
)
[C#]
protected override void Dispose(
   bool disposing
);
[C++]
protected: void Dispose(
   bool disposing
);
[JScript]
protected override function Dispose(
   disposing : Boolean
);
```

**Parameters**

**disposing**

**Remarks**

This method is called by the public Dispose method and the Finalize method. Dispose() invokes the protected Dispose(Boolean) method with the **disposing** parameter set to `true`. Finalize invokes Dispose with **disposing** set to `false`.

When the **disposing** parameter is `true`, the method releases all resources held by any managed objects that this IfxCommand references. It does this by invoking the Dispose() method of each referenced object.

**Notes to inheritors:** Dispose can be called multiple times by other objects. When overriding Dispose(Boolean), be careful not to reference objects that have been previously disposed of in an earlier call to Dispose. For more information about how to implement Dispose(Boolean), see "Implementing a Dispose Method" in the Microsoft(R) .NET Framework SDK documentation.

Calling Dispose on a IfxConnection object is different from calling Close. For example, Dispose clears the connection string while Close does not. For more information about Dispose and Finalize, see "Cleaning Up Unmanaged Resources," and "Overriding the Finalize Method," in the .NET Framework SDK documentation.

**Example**

[Visual Basic, C#] The following example creates a IfxConnection and then disposes of it.

```
[Visual Basic]
Public Sub IfxConnectionHereAndGone()
    Dim myConnection As New IfxConnection()
    myConnection.Open()
    'Calling Dispose also calls IfxConnection.Close.
    myConnection.Dispose()
End Sub

[C#]
public void IfxConnectionHereAndGone()
{
   IfxConnection myConnection = new IfxConnection();
   myConnection.Open();
   //Calling Dispose also calls IfxConnection.Close.
   myConnection.Dispose();
}
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

"IfxConnection.Dispose Method" on page 5-178

**IfxConnection.DropDTD Method:**

Drops the DTD registered with the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Parameters**

**Return value**

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnection Class" on page 5-157

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.EndChain Method:**

Marks the end of a chain of insert, update, and delete statements to be sent to the database server.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub EndChain()
[C#]
public void EndChain();
[C++]
public: void EndChain();
[JScript]
public function EndChain();
```

**Remarks**

To execute all the statements accumulated since the call to the BeginChain method, call the EndChain method.

**Example**

[Visual Basic, C#] The following example uses chaining to insert 10000 rows into the STAFF table.

```
[Visual Basic]
Dim con As IfxConnection = new IfxConnection("DATABASE=sample;")
Dim cmd As IfxCommand = con.CreateCommand()
con.Open()

' Initialize an insert statement using parameter markers
cmd.CommandText = "INSERT INTO STAFF(ID) VALUES( ? )"

' Add a parameter
Dim p1 As IfxParameter = cmd.Parameters.Add("@ID", IfxType.Integer )

' Start the chain
con.BeginChain()

Try
   ' Loop to add 10000 rows
   Dim I As Int32
   For I = 1 To 10000
      ' Set the parameter value
```

```
      p1.Value = I

      ' Execute the command.
      ' Since chaining is active, this statement is now added
      '   to the chain
      cmd.ExecuteNonQuery()
   Next I

   ' Execute the chain
   con.EndChain()
Catch db2Ex As IfxException
   Dim db2Error As IfxError

   ' Loop through all the errors
   For Each db2Error in db2Ex.Errors
      Console.WriteLine("SQLSTATE =" & db2Error.SQLState )
      Console.WriteLine("NativeErr=" & db2Error.NativeError )
      Console.WriteLine("RowNumber=" & db2Error.RowNumber )
      Console.WriteLine( db2Error.Message )
   Next IfxError
Finally
   ' Explicitly turn chaining off in case it is still on
   If (con.Chaining) Then
      con.EndChain()
   End If
End Try

con.Close()[C#]
DB2Connection con = new IfxConnection("DATABASE=sample;");
DB2Command cmd = con.CreateCommand();
con.Open();

// Initialize an insert statement using parameter markers
cmd.CommandText = "INSERT INTO STAFF(ID) VALUES( ? )";

// Add a parameter
DB2Parameter p1 = cmd.Parameters.Add("@ID", IfxType.Integer );

// Start the chain
con.BeginChain();

try
{
   // Loop to add 10000 rows
   for( Int32 i = 1; i <= 10000; i++ )
   {
      // Set the parameter value
      p1.Value = i;

      // Execute the command.
      // Since chaining is active, this statement is now added
      //   to the chain
      cmd.ExecuteNonQuery();
   }

   // Execute the chain
   con.EndChain();
}
catch( IfxException db2Ex )
{
   // Loop through all the errors
   foreach( IfxError db2Error in db2Ex.Errors )
   {
      Console.WriteLine("SQLSTATE =" + db2Error.SQLState );
      Console.WriteLine("NativeErr=" + db2Error.NativeError );
      Console.WriteLine("RowNumber=" + db2Error.RowNumber );
      Console.WriteLine( db2Error.Message );
```

```
   }
}
finally
{
   // Explicitly turn chaining off in case it is still on
   if( con.Chaining )
   {
      con.EndChain();
   }
}

con.Close();
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

"IfxConnection.BeginChain Method" on page 5-167

### IfxConnection.EnlistDistributedTransaction Method:

Enlists in the specified transaction as a distributed transaction.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub EnlistDistributedTransaction( _
   ByVal transaction As ITransaction _
)
[C#]
public void EnlistDistributedTransaction(
   ITransaction transaction
);
[C++]
public: void EnlistDistributedTransaction(
   ITransaction* transaction
);
[JScript]
public function EnlistDistributedTransaction(
   transaction : ITransaction
);
```

**Parameters**

**transaction**
A reference to an existing transaction in which to enlist.

**Remarks**

You can enlist in an existing distributed transaction using the EnlistDistributedTransaction method if auto-enlistment is disabled. Enlisting in an existing distributed transaction ensures that, if the transaction is committed or rolled back, modifications made by the code at the data source are also committed or rolled back.

**CAUTION:**
**EnlistDistributedTransaction returns an exception if the IfxConnection has already started a transaction using BeginTransaction . However, if the transaction is a local transaction started at the data source (for example, by explicity executing the BEGIN TRANSACTION statement using a IfxCommand object), EnlistDistributedTransaction rolls back the local transaction and enlists in the existing distributed transaction as requested. You will not receive notice that the local transaction was rolled back, and are responsible for managing any local transactions not started using BeginTransaction.**

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.GetDTD Method:**

Gets the DTD registered with the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Parameters**

**Return value**

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnection Class" on page 5-157

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.GetSchema Method:**

Returns a DataTable with metadata for the data source associated with this IfxConnection instance.

**Overload list**

| Name | Description |
|---|---|
| GetSchema() | Returns a DataTable with metadata for the data source associated with this IfxConnection instance. |
| GetSchema(String) | Returns a DataTable with a specified set of metadata for the data source associated with this IfxConnection instance. |
| GetSchema(String, String[]) | Returns a DataTable with a specified set of metadata for the data source associated with this IfxConnection instance. |

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

*IfxConnection.GetSchema () Method:*

Returns a DataTable that contains metadata for the data source that is associated with an IfxConnection instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function GetSchema As DataTable
[C#]
public override DataTable GetSchema ()
[C++]
public:
virtual DataTable^ GetSchema () override
[JScript]
public override function GetSchema () : DataTable
```

**Return value**

A DataTable instance, that contains metadata for the data source that is associated with an IfxConnection instance.

**Example**

[C#] The following line of C# code demonstrates how to retrieve the available metadata for the data source that is associated with an IfxConnection instance. The application can use the module name as one of the restrictions.

```
[C#]
    DataTable table2 = conn.GetSchema();
```

Example of values returned by the GetSchema method are as follows:

```
TABLE_SCHEMA=ADMINISTRATOR
TABLE_SCHEMA=NULLID
-
-
TABLE_SCHEMA=SYSSTAT
```

**Version information**

**.NET Framework version**
Supported in: 2.0 , 3.0, 3.5, and 4.0

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

*IfxConnection.GetSchema (String) Method:*

Returns a DataTable with a specified set of metadata for the data source associated with this IfxConnection instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function GetSchema ( _
   collectionName As String _
) As DataTable
[C#]
public override DataTable GetSchema (
   string collectionName
);
[C++]
public:
virtual DataTable^ GetSchema (
   String collectionName
) override
[JScript]
public override function GetSchema (
   collectionName : String
) : DataTable
```

**Parameters**

**collectionName**
Specifies the set of data source metadata to return. The following table lists read-only string properties you can use to identify the metadata collection GetSchema will return.

| Collection name | Description |
|---|---|
| IfxMetaDataCollectionNames.MetaDataCollections | A list of the metadata collections supported by the IBM Data Provider for .NET. |
| IfxMetaDataCollectionNames.Restrictions | For each metadata collection, a list of qualifiers that can be used to restrict the scope of the requested metadata. |
| IfxMetaDataCollectionNames.DataSourceInformation | Information about the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.DataTypes | A list of all the data types supported by DB2 family databases. |
| IfxMetaDataCollectionNames.ReservedWords | A list of all the reserved words for the DB2 family SQL dialect. |
| IfxMetaDataCollectionNames.Tables | A list of the tables in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.TablePrivileges | A list of the table privileges in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.Columns | A list of the table columns in the data source associated with this IfxConnection instance. |

| Collection name | Description |
|---|---|
| IfxMetaDataCollectionNames.ColumnPrivileges | A list of the column privileges in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.Procedures | A list of the stored procedures in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.ProcedureParameters | A list of the stored procedure parameters in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.PrimaryKeys | A list of the table primary keys in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.ForeignKeys | A list of the table foreign keys in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.Indexes | A list of the indexes in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.Schemas | A list of the schemas in the data source associated with this IfxConnection instance. |

**Return value**

A DataTable instance, which contains information about the visible DB2 family databases.

**Example**

[C#] The following line of code demonstrates how to retrieve the available metadata for the data source associated with this IfxConnection instance. The use of this particular overload of GetSchema (with a String parameter) results in the retrieval of a list of the stored procedures cataloged in the data source.

```
[C#]
    DataTable proctable = conn.GetSchema(IfxMetaDataCollectionNames.Procedures);
```

**Version information**

**.NET Framework version**

Supported in: 2.0 and 3.0

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

*IfxConnection.GetSchema (String, String[]) Method:*

Returns a DataTable with a specified set of metadata for the data source associated with this IfxConnection instance.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Overrides Function GetSchema ( _
   collectionName As String, _
   restrictionValues As String() _
) As DataTable
[C#]
public override DataTable GetSchema (
   string collectionName,
   string[] restrictionValues
);
[C++]
public:
virtual DataTable^ GetSchema (
   String collectionName,
   array<String^>^ restrictionValues
) override
[JScript]
public override function GetSchema (
   collectionName : String,
   restrictionValues : String[]
) : DataTable
```

### Parameters

**collectionName**

Specifies the set of data source metadata to return. The following table lists read-only string properties you can use to identify the metadata collection GetSchema will return.

| Collection name | Description |
|---|---|
| IfxMetaDataCollectionNames.MetaDataCollections | A list of the metadata collections supported by the IBM Data Provider for .NET. |
| IfxMetaDataCollectionNames.Restrictions | For each metadata collection, a list of qualifiers that can be used to restrict the scope of the requested metadata. |
| IfxMetaDataCollectionNames.DataSourceInformation | Information about the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.DataTypes | A list of all the data types supported by DB2 family databases. |
| IfxMetaDataCollectionNames.ReservedWords | A list of all the reserved words for the DB2 family SQL dialect. |
| IfxMetaDataCollectionNames.Tables | A list of the tables in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.TablePrivileges | A list of the table privileges in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.Columns | A list of the table columns in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.ColumnPrivileges | A list of the column privileges in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.Procedures | A list of the stored procedures in the data source associated with this IfxConnection instance. |

| Collection name | Description |
|---|---|
| IfxMetaDataCollectionNames.ProcedureParameters | A list of the stored procedure parameters in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.PrimaryKeys | A list of the table primary keys in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.ForeignKeys | A list of the table foreign keys in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.Indexes | A list of the indexes in the data source associated with this IfxConnection instance. |
| IfxMetaDataCollectionNames.Schemas | A list of the schemas in the data source associated with this IfxConnection instance. |

**restrictionValues**

You can refine the set of metadata returned by the GetSchema method by assigning values for the restrictions specific to the applicable metadata collection. Pass these restrictions values into the GetSchema method as a String array. To see the list of restrictions for each metadata collection, read the DataTable generated by the following code:

```
DataTable resttable =
conn.GetSchema(IfxMetaDataCollectionNames.Restrictions);
```

**Return value**

A DataTable instance, which contains information about the visible DB2 family databases.

**Example**

[C#] The following line of code demonstrates how to retrieve the available metadata for the data source associated with this IfxConnection instance. The use of this particular overload of GetSchema (with a String and a String array parameters) results in the retrieval of a list of the tables in the data source associated with the ERIK schema.

```
[C#]
DataTable eriktab = conn.GetSchema(IfxMetaDataCollectionNames.Tables,
                                   new string[4] { null, "ERIK", null, null } );
```

**Version information**

**.NET Framework version**

Supported in: 2.0 and 3.0

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.Open Method:**

Opens a connection to a database with the property settings specified by the ConnectionString.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Public Sub Open() Implements IDbConnection.Open
[C#]
public void Open();
[C++]
public: __sealed void Open();
[JScript]
public function Open();
```

**Implements:**
IDbConnection.Open

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidOperationException | The connection is already open. |
| IfxException | A connection-level error occurred while opening the connection. |

**Remarks**

The IfxConnection.Open Method supports the System.EnterpriseServices namespace and the System.Transactions namespace.

The IfxConnection draws an open connection from the connection pool if connection pooling is on and a connection is available. Otherwise, it establishes a new connection to the database.

**Note:** If the IfxConnection goes out of scope, the connection it represents does not close automatically. Therefore, you must explicitly close the connection by calling Close or Dispose.

**Example**

[Visual Basic, C#] The following example creates a IfxConnection , opens it, displays some of its properties, then closes the connection.

```
[Visual Basic]
Public Sub CreateIfxConnection(myConnString As String)
    Dim myConnection As New IfxConnection(myConnString)
    myConnection.Open()
    MessageBox.Show("ServerVersion: " + myConnection.ServerVersion _
       + ControlChars.Cr + "State: " + myConnection.State.ToString())
    myConnection.Close()
End Sub

[C#]
public void CreateIfxConnection(string myConnString)
{
   IfxConnection myConnection = new IfxConnection(myConnString);
   myConnection.Open();
```

```
   MessageBox.Show("ServerVersion: " + myConnection.ServerVersion
      + "\nState: " + myConnection.State.ToString());
   myConnection.Close();
}
```

In DB2 for z/OS Version 10, if the server has migrated to a different mode during the application execution and if a call to Open has returned a connection with migrated server info, then subsequent calls to Open will always return a connection with migrated server info. All connections established prior to the migration will be recycled on the call to Close.

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

"IfxConnection.Close Method" on page 5-176

### IfxConnection.RegisterDTD Method:

Registers the DTD with the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Parameters**

**Return value**

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnection Class" on page 5-157

"IBM.Data.Informix Namespace" on page 5-1

### IfxConnection.ReleaseObjectPool Method:

Indicates that the resources of the connection pool can be released when the last underlying connection is released.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Sub ReleaseObjectPool()
[C#]
public static void ReleaseObjectPool();
[C++]
public: static void ReleaseObjectPool();
[JScript]
public static function ReleaseObjectPool();
```

**Remarks**

ReleaseObjectPool can be called to release the resources of the connection pool. You can call this method if, for example, the connection object will not be used again. When all connections in the pool are closed, the pool can be disposed of. Note that calling the method alone does not actually release the active connections that exist in the pool.

The following must occur before the pool is finally disposed:

1. Call Close to release the IfxConnection object from the environment.
2. Allow each connection object to time out.
3. Call ReleaseObjectPool.
4. Invoke garbage collection.

Conversely, if you call Close on all active connections, and invoke garbage collection, but do not call ReleaseObjectPool, the resources reserved for the pool will remain available.

After a pool is released, a request for a new IfxConnection creates a new pool.

**Example**

[Visual Basic, C#] The following example creates a IfxConnection, opens it, displays some of its properties, closes the connection and releases the object pool to conserve resources.

```
[Visual Basic]
Public Sub CreateIfxConnection(myConnString As String)
     Dim myConnection As New IfxConnection(myConnString)
     myConnection.Open()
     MessageBox.Show("State: " + myConnection.State.ToString())
     myConnection.Close()
     IfxConnection.ReleaseObjectPool()
 End Sub

[C#]
public void CreateIfxConnection(string myConnString)
 {
    IfxConnection myConnection = new IfxConnection(myConnString);
    myConnection.Open();
    MessageBox.Show("State: " + myConnection.State.ToString());
    myConnection.Close();
    IfxConnection.ReleaseObjectPool();
 }
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

## IfxConnection Properties

The properties of the IfxConnection class are listed here. For a complete list of IfxConnection class members, see the IfxConnection Members topic.

## Public Properties

| Name | Description |
|---|---|
| CacheData | Gets or sets a boolean value indicating whether data caching is being used. |
| Chaining | Gets a boolean value indicating whether chaining is active. |
| ClientAccountingInformation | Gets or sets the string containing the client accounting information. |
| ClientApplicationInformation | Gets or sets the string containing the client application name. |
| ClientUser | Gets or sets the string that contains the client user ID. |
| ClientWorkstation | Gets or sets the string that contains the name of the client workstation. |
| ConnectionString | Gets or sets the string that is used to open a database connection. |
| Container (inherited from Component) | Gets the IContainer that contains the Component. |
| ConnectionTimeout | Gets or sets a value that determines the limit to the amount of time that an application waits for a connection. |
| Database | Gets the name of the current database or the database to be used after a connection is opened. |
| DataSource | Gets the name of the current database or the database to be used after a connection is opened. |
| ResultSetAsReturnValue | Set this value to work with IDS UDRs that return result sets. |
| ServerBuildVersion | Gets a string containing the build version of the server to which the client is connected. |
| ServerMajorVersion | Gets a string containing the major version of the server to which the client is connected. |
| ServerMinorVersion | Gets a string containing the minor version of the server to which the client is connected. |
| ServerRevisionVersion | Gets a string containing the revision version of the server to which the client is connected. |
| ServerType | Gets a string containing the type of server to which the client is connected. |
| ServerVersion | Gets a string containing the version of the server to which the client is connected. |
| Site (inherited from Component) | Gets or sets the ISite of the Component. |
| State | Gets the current state of the connection. |

## Protected Properties

| Property | Description |
|---|---|
|  DesignMode (inherited from Component) | Gets a value that indicates whether the Component is in design mode. |
|  Events (inherited from Component) | Gets the list of event handlers that are attached to a Component. |

**Reference**

"IfxConnection Class" on page 5-157

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.CacheData Property:**

Gets or sets a boolean value indicating if data caching is being used.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property CacheData As Boolean
[C#]
public bool CacheData {get;}
[C++]
public: __property bool get_CacheData();
[JScript]
public function get CacheData() : Boolean;
```

**Property value**

True if chaining is active; otherwise false.

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.Chaining Property:**

Gets a boolean value indicating if chaining is active.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Chaining As Boolean
[C#]
public bool Chaining {get;}
```

```
[C++]
public: __property bool get_Chaining();
[JScript]
public function get Chaining() : Boolean;
```

**Property value**

True if chaining is active. Chaining is active for a IfxConnection object if the application has called that object's BeginChain method, but has not yet called its EndChain method.

False if chaining is not active.

**Example**

[Visual Basic, C#] The following example uses chaining to insert 10000 rows into the STAFF table.

```
[Visual Basic]
Dim con As IfxConnection = new IfxConnection("DATABASE=sample;")
Dim cmd As IfxCommand = con.CreateCommand()
con.Open()

' Initialize an insert statement using parameter markers
cmd.CommandText = "INSERT INTO STAFF(ID) VALUES( ? )"

' Add a parameter
Dim p1 As IfxParameter = cmd.Parameters.Add("@ID", IfxType.Integer )

' Start the chain
con.BeginChain()

Try
   ' Loop to add 10000 rows
   Dim I As Int32
   For I = 1 To 10000
      ' Set the parameter value
      p1.Value = I

      ' Execute the command.
      ' Since chaining is active, this statement is now added
      '   to the chain
      cmd.ExecuteNonQuery()
   Next I

   ' Execute the chain
   con.EndChain()
Catch db2Ex As IfxException
   Dim db2Error As IfxError

   ' Loop through all the errors
   For Each db2Error in db2Ex.Errors
      Console.WriteLine("SQLSTATE =" & db2Error.SQLState )
      Console.WriteLine("NativeErr=" & db2Error.NativeError )
      Console.WriteLine("RowNumber=" & db2Error.RowNumber )
      Console.WriteLine( db2Error.Message )
   Next IfxError
Finally
   ' Explicitly turn chaining off in case it is still on
   If (con.Chaining) Then
      con.EndChain()
   End If
End Try

con.Close()[C#]
```

```
DB2Connection con = new IfxConnection("DATABASE=sample;");
DB2Command cmd = con.CreateCommand();
con.Open();

// Initialize an insert statement using parameter markers
cmd.CommandText = "INSERT INTO STAFF(ID) VALUES( ? )";

// Add a parameter
DB2Parameter p1 = cmd.Parameters.Add("@ID", IfxType.Integer );

// Start the chain
con.BeginChain();

try
{
   // Loop to add 10000 rows
   for( Int32 i = 1; i <= 10000; i++ )
   {
      // Set the parameter value
      p1.Value = i;

      // Execute the command.
      // Since chaining is active, this statement is now added
      //   to the chain
      cmd.ExecuteNonQuery();
   }

   // Execute the chain
   con.EndChain();
}
catch( IfxException db2Ex )
{
   // Loop through all the errors
   foreach( IfxError db2Error in db2Ex.Errors )
   {
      Console.WriteLine("SQLSTATE =" + db2Error.SQLState );
      Console.WriteLine("NativeErr=" + db2Error.NativeError );
      Console.WriteLine("RowNumber=" + db2Error.RowNumber );
      Console.WriteLine( db2Error.Message );
   }
}
finally
{
   // Explicitly turn chaining off in case it is still on
   if( con.Chaining )
   {
      con.EndChain();
   }
}

con.Close();
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

"IfxConnection.BeginChain Method" on page 5-167

### IfxConnection.ClientAccountingInformation Property:

Gets or sets the string containing the client accounting string.

**Namespace:**

IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ClientAccountingInformation As String
[C#]
public string ClientAccountingInformation {get; set;}
[C++]
public: __property String* get_ClientAccountingInformation();
public: __property void set_ClientAccountingInformation(String*);
[JScript]
public function get ClientAccountingInformation() : String;
public function set ClientAccountingInformation(String);
```

**Property value**

The client accounting string. The default value is an empty string (""). This value changes only when the application explicitly updates it.

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.ClientApplicationInformation Property:**

Gets or sets the string containing the client application name.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ClientApplicationInformation As String
[C#]
public string ClientApplicationInformation {get; set;}
[C++]
public: __property String* get_ClientApplicationInformation();
public: __property void set_ClientApplicationInformation(String*);
[JScript]
public function get ClientApplicationInformation() : String;
public function set ClientApplicationInformation(String);
```

**Property value**

The client application name. The default value is an empty string (""). This value changes only when the application explicitly updates it.

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.ClientUser Property:**

Gets or sets the string containing the client user ID.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ClientUser As String
[C#]
public string ClientUser {get; set;}
[C++]
public: __property String* get_ClientUser();
public: __property void set_ClientUser(String*);
[JScript]
public function get ClientUser() : String;
public function set ClientUser(String);
```

**Property value**

The client user ID. The default value is an empty string (""). This value changes only when the application explicitly updates it.

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.ClientWorkStation Property:**

Gets or sets the string containing the name of the client workstation.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ClientWorkStation As String
[C#]
public string ClientWorkStation {get; set;}
[C++]
public: __property String* get_ClientWorkStation();
public: __property void set_ClientWorkStation(String*);
[JScript]
public function get ClientWorkStation() : String;
public function set ClientWorkStation(String);
```

**Property value**

The name of the client workstation. The default value is an empty string (""). This value changes only when the application explicitly updates it.

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

### IfxConnection.ConnectionString Property:

Gets or sets the string used to open a database connection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ConnectionString As String  Implements _
   IDbConnection.ConnectionString
[C#]
public string ConnectionString {get; set;}
[C++]
public: __property String* get_ConnectionString();
public: __property void set_ConnectionString(String*);
[JScript]
public function get ConnectionString() : String;
public function set ConnectionString(String);
```

**Implements:**
IDbConnection.ConnectionString

**Property value**

The connection string that includes settings, such as the database name, that are needed to establish an initial connection. The default value is an empty string (""). The maximum length is 1024 characters.

**Remarks**

You can use the ConnectionString property to connect to any supported data server. You can set the ConnectionString property only when the connection is closed.

The supported keywords are as follows.

**Important:** Only pureQuery keywords are case sensitive.

*Table 5-15. Common keywords*

| Keyword | Default | Description |
|---|---|---|
| **Database** | | Database alias (for a cataloged database). |
| **Password** \| **PWD** | | Password |
| **Server** | | Server name with optional port number for a direct connection using either IPv4 notation (*server name*\|*ip address*[:*port*]) or IPv6 notation. |

*Table 5-16. Additional keywords*

| Keyword | Default | Description |
|---|---|---|
| **Authentication** | SERVER | Type of authentication. Acceptable values are as follows:<br>• SERVER<br>• SERVER_ENCRYPT<br>• DATA_ENCRYPT<br>• KERBEROS<br>• GSSPLUGIN |
| **ClientAccountingString** | | Client accounting string that is sent to a database. |
| **ClientApplicationName** | | Client application name that is sent to a database. |
| **CodePage** | 0 | Code page identifier. |
| **ConcurrentAccessResolution** | | String values representing the concurrent access resolution. Acceptable values are as follows:<br>• CurrentlyCommitted – Allows to use the currently committed version of the data that is updated or deleted on a PREPAREstatement.<br>• WaitForOutcome – Allows the COMMIT or ROLLBACK action to wait until the updates to a PREPARE statement.<br>• SkipLockedData – Allows to skip locks on uncommitted inserts and rows that are not updated on a PREPARE statement. |
| **Connection Lifetime** | 60 | Amount of time in seconds that a connection can remain idle in the connection pool. |
| **Connection Reset** | false | Indication of whether a connection is put into the connection pool when the connection is closed. Acceptable values are as follows:<br>• true. The connection is not put into the connection pool when the connection is closed.<br>• false. The connection is put into the connection pool when the connection is closed. |
| **Connection Timeout** \| **Connect Timeout** \| **Timeout** | 0 | Time in seconds to wait for a database connection to be established. A value of 0 indicates there is no time limit for a database connection to be established. If you enable client reroute, the maximum waiting time is approximately double the specified connection timeout value because there is a second attempt to connect to the alternative server. |
| **CurrentSchema** | | Schema name that is used to qualify all unqualified SQL objects that are used with the current connection. |
| **DB2Explain** | off | The acceptable values are as follows:<br>• Off– Disables the Explain information for snapshot and tables.<br>• Table– Generates Explain information for table.<br>• Snapshot – Generates Explain information for snapshot.<br>• SnapshotAndTable– Enables the Explain information for snapshot and tables. |
| **DisableCursorHold** | false | Indication of whether cursors opened on the server should be left open after committing a transaction. Acceptable values are as follows:<br>• true. Cursors opened on the server are closed after the transaction is committed.<br>• false. Cursors opened on the server are left open after the transaction is committed. |

*Table 5-16. Additional keywords (continued)*

| Keyword | Default | Description |
|---|---|---|
| **Enlist** | true | Indication of whether enlistment in the Distributed Transaction Coordinator (DTC) is allowed. Acceptable values are as follows:<br>• true. Enlistment in the DTC is allowed. Enlistment occurs only if a COM+ transaction is in progress at connection time.<br>• false. Enlistment in the DTC is not allowed. |
| **FitHighPrecisionType** | RetExc* | **Note:** *- ReturnExceptionThe acceptable values are as follows:<br>• WithTruncate. Results in returning a .NET system type after silently truncating the column value if needed.<br>• AsString. Results in converting the column to a .NET string type.<br>• ReturnException. Results in a truncation exception if the value does not fit in the .NET system type. |
| **HostVarParameter** | false | Indication of whether host variable (:param) support is enabled. Acceptable values are as follows:<br>• true. Host variable support is enabled.<br>• false. Host variable support is disabled. |
| **Interrupt** | 1 | Interrupt processing mode. Acceptable values are as follows:<br>• 0. Interrupt processing is disabled.<br>• 1. If the server supports an interrupt, an interrupt is sent. Otherwise, the connection is dropped.<br>• 2. An interrupt results in a dropped connection, regardless of the interrupt capabilities of the server. |
| **Isolation Level** \| **IsolationLevel** | | Isolation level for the connection. Acceptable values are as follows:<br>• ReadCommitted<br>• ReadUncommitted<br>• RepeatableRead<br>• Serializable<br>• Transaction<br><br>This keyword is supported only for applications participating in a distributed transaction, such as a COM+ application. If you use this keyword with applications that do not participate in a distributed transaction, an invalid argument exception is thrown.<br><br>If you set the isolation level to Transaction, the isolation level is set to the value of Transaction.Current.IsolationLevel. |
| **Max Pool Size** | No maximum | Maximum pool size. |
| **Min Pool Size** | 0 | Minimum pool size. |
| **Persist Security Info** | false | Indication of whether security-sensitive information is returned. Acceptable values are as follows:<br>• true. Allow security-sensitive information, such as a password, to be returned as part of a connection string after the connection has been opened or if the connection has ever been open.<br>• false. Do not return security-sensitive information as part of a connection string. This value is strongly recommended. |

*Table 5-16. Additional keywords (continued)*

| Keyword | Default | Description |
|---|---|---|
| **Pooling** | true | Indication of whether connection pooling is disabled. Acceptable values are as follows:<br>• true. Connection pooling is enabled.<br>• false. Connection pooling is disabled. |
| **ProgramId** | | Program ID that is sent to a database. |
| **ProgramName** | | Program name that is sent to a database. |
| **ResultArrayAsReturnValue** | false | Indication of whether results sets from Informix user-defined routines (UDRs) are returned as parameters of type ReturnValue.<br>If you set the **ResultArrayAsReturnValue** keyword to true, result sets are returned as parameters of ReturnValue.This keyword is ignored when a ReturnValue parameter is not bound in the parameter collection. |
| **RetrieveXMLInBinaryFormat** | false | The value representing whether or not XML data should be returned in binary format. |
| **Security** | | Indication of whether to use SSL as a secure transport. If you set this keyword to SSL, SSL is used. |
| **SessionTimeZone** | | Sets the value of the SESSION TIMEZONE special register on the z/OS server. |
| **StatementConcentrator** | | Indication of whether statement concentration is used. Acceptable values are as follows:<br>• Off. The statement concentrator is disabled.<br>• Literals. The statement concentrator with literals behavior, is enabled. Server-side statement concentrator restrictions apply.<br>If you do not set this keyword, the literal replacement behavior is determined by the server configuration. |
| **TrustedContextSystemUserID \| TCSUID** | | Trusted context SYSTEM AUTHID to be used with the connection. |
| **TrustedContextSystemPassword \| TCSPWD** | | Password corresponding to the trusted context SYSTEM AUTHID to be used with the connection. |

Many of the settings that you can specify in the string have corresponding read-only properties, for example, the **DATABASE** corresponds to the Database property. When you set the connection string, all of these read-only properties are updated, except if an error is detected. In this case, none of the properties are updated. IfxConnection properties, such as Database) return only default settings or those settings that you specified for the ConnectionString property.

Validation of the connection string occurs when you set it.

Resetting the ConnectionString property on a closed connection resets all connection string values and related properties, including the password.

**Example**

[Visual Basic, C#] The following examples create an IfxConnection and set some of its properties in the connection string:

```
[Visual Basic]
Public Sub CreateIfxConnection()
    Dim myConnString As String = _
```

```
       "DATABASE=SAMPLE;"
    Dim myConnection As New IfxConnection(myConnString)
    myConnection.Open()
    MessageBox.Show("ServerVersion: " + myConnection.ServerVersion _
       + ControlChars.Cr + "Database: " + myConnection.Database.ToString())
    myConnection.Close()
End Sub

[C#]
public void CreateIfxConnection()
{
   string myConnString = "DATABASE=SAMPLE;";
   IfxConnection myConnection = new IfxConnection(myConnString);
   myConnection.Open();
   MessageBox.Show("ServerVersion: " + myConnection.ServerVersion
      + "\nDatabase: " + myConnection.Database.ToString());
   myConnection.Close();
}
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

### IfxConnection.ConnectionTimeout Property:

Gets a value that determines the limit to the amount of time that an application should wait for a connection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### .NET Framework 2.0 3.0, 3.5, and 4.0 Syntax

```
[Visual Basic]
Overrides Public ReadOnly Property ConnectionTimeout As Integer
[C#]
public override int ConnectionTimeout {get;}
[C++]
public: __property virtual int get_ConnectionTimeout();
[JScript]
public function get ConnectionTimeout() : int;
```

### Property value

The time (in seconds) to wait while trying to establish a connection before terminating the attempt and reporting an error. The default value is 0, which indicates there is no time limit for the database connection to be established.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentException | The value set is less than 0. |

### Remarks

If client re-route is enabled, the maximum waiting time will approximately double that of the specified connect timeout, because there will be second connection attempt to the alternate server.

### Example

[Visual Basic, C#] The following example creates a IfxConnection using the connection time keyword, and then checks the ConnectionTime

```
[Visual Basic]
Public Sub CreateIfxConnection()
    Dim conn As New IfxConnection("Database=SAMPLE;Connect Timeout=30")
    conn.Open()
    Dim timeout As Integer = conn.ConnectionTimeout
End Sub

[C#]
public void CreateIfxConnection()
{
   IfxConnection conn = new IfxConnection("Database=SAMPLE;Connect Timeout=30");
   conn.Open();
   int timeout = conn.ConnectionTimeout;
}
```

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxConnection.ConnectionString Property" on page 5-199

"IfxConnection Class" on page 5-157

"IBM.Data.Informix Namespace" on page 5-1

## IfxConnection.DataSource Property:

Gets the name of the current database or the database to be used after a connection is opened.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Overrides ReadOnly Property DataSource As String
[C#]
public override string DataSource { get; }
[C++]
public:
virtual property String^ DataSource {
 String^ get () override;
}
[JScript]
public override function get DataSource () : String
```

**Property value**

The name of the current database. The default value is an empty string ("") until the connection is opened.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.Database Property:**

Gets the name of the current database or the database to be used after a connection is opened.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Database As String  Implements _
   IDbConnection.Database
[C#]
public string Database {get;}
[C++]
public: __property String* get_Database();
[JScript]
public function get Database() : String;
```

**Implements:**
IDbConnection.Database

**Property value**

The name of the current database. The default value is an empty string ("") until the connection is opened.

**Remarks**

Initially, the Database property is set in the connection string. The Database property can be updated by using the ChangeDatabase method. If you change the current database by using the ChangeDatabase method, an informational message is sent and then the property is updated.

**Example**

[Visual Basic, C#] The following example creates a IfxConnection and changes the current database.

```
[Visual Basic]
Public Sub CreateIfxConnection()
     Dim myConnString As String = _
        "DATABASE=SAMPLE"
```

```
    Dim myConnection As New IfxConnection(myConnString)
    myConnection.Open()
    myConnection.ChangeDatabase("SAMPLE2")
    MessageBox.Show("Database: " + myConnection.Database.ToString())
    myConnection.Close()
 End Sub
[C#]
public void CreateIfxConnection()
 {
    string myConnString = "DATABASE=SAMPLE";
    IfxConnection myConnection = new IfxConnection(myConnString);
    myConnection.Open();
    myConnection.ChangeDatabase("SAMPLE2");
    MessageBox.Show("Database: " + myConnection.Database.ToString());
    myConnection.Close();
 }
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

"IfxConnection.ChangeDatabase Method" on page 5-175

"IfxConnection.ConnectionString Property" on page 5-199

### IfxConnection.ResultSetAsReturnValue Property:

Set this value to retrieve result sets from Informix user-defined routines (UDR) as a ReturnValue parameter. This property is ignored when a ReturnValue parameter is not bound in the parameter collection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ResultSetAsReturnValue As Boolean
[C#]
public bool ResultSetAsReturnValue {get; set;}
[C++]
public: __property bool get_ResultSetAsReturnValue();
public: __property void set_ResultSetAsReturnValue(bool);
[JScript]
public function get ResultSetAsReturnValue() : Boolean;
public function set ResultSetAsReturnValue(Boolean);
```

**Property value**

Set to `true` to work with Informix UDRs that return result sets. Otherwise, set to `false`. The default value is `false`.

**Exceptions**

| Exception type | Condition |
|---|---|
| ArguementException | Used on an unsupported server. |

**Remarks**

If this property is set, and theUDR returns multiple types or multiple rows, the ReturnValue parameter contains the value of the first row and first column. Other values are ignored.

This property does not affect IfxConnection.DeriveParameters(). Applications should specify the correct type for the ReturnValue parameter. If the type does not match the ReturnValue parameter, an exception is generated.

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.ServerMajorVersion Property:**

Gets an integer containing the major version number of the server to which the client is connected.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property ServerMajorVersion As Integer
[C#]
public int ServerMajorVersion {get;}
[C++]
public: __property int get_ServerMajorVersion();
[JScript]
public function get ServerMajorVersion() : int;
```

**Property value**

Gets an integer containing the major version number of the server to which the client is connected. The default value is an empty string ("") until the connection is opened.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.ServerBuildVersion Property:**

Gets an integer containing the build version number of the server to which the client is connected.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property ServerBuildVersion As Integer
[C#]
public int ServerBuildVersion {get;}
[C++]
public: __property int get_ServerBuildVersion();
[JScript]
public function get ServerBuildVersion() : int;
```

**Property value**

Gets an integer containing the build version number of the server to which the client is connected. The default value is an empty string ("") until the connection is opened.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.ServerMinorVersion Property:**

Gets an integer containing the minor version number of the server to which the client is connected.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property ServerMinorVersion As Integer
[C#]
public int ServerMinorVersion {get;}
[C++]
public: __property int get_ServerMinorVersion();
[JScript]
public function get ServerMinorVersion() : int;
```

**Property value**

Gets an integer containing the minor version number of the server to which the client is connected. The default value is an empty string ("") until the connection is opened.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.ServerRevisionVersion Property:**

Gets an integer containing the revision version number of the server to which the client is connected.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property ServerRevisionVersion As Integer
[C#]
public int ServerRevisionVersion {get;}
[C++]
public: __property int get_ServerRevisionVersion();
[JScript]
public function get ServerRevisionVersion() : int;
```

**Property value**

Gets an integer containing the revision version number of the server to which the client is connected. The default value is an empty string ("") until the connection is opened.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.ServerType Property:**

Gets a string containing the type of the server to which the client is connected.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property ServerType As String
[C#]
public string ServerType {get;}
[C++]
```

```
public: __property String* get_ServerType();
[JScript]
public function get ServerVersion() : String;
```

**Property value**

The type of the connected server. The default value is an empty string ("") until the connection is opened.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidOperationException | The connection is closed. |

**Remarks**

The ServerType property is a string that is specific to the data server type for this connection.

**Example**

[Visual Basic, C#] The following example creates a IfxConnection and displays the ServerType.

```
[Visual Basic]
Public Sub CreateIfxConnection()
     Dim myConnectString As String = "DATABASE=SAMPLE;"
     Dim myConnection As New IfxConnection(myConnectString)
     myConnection.Open()
     MessageBox.Show("ServerType: " + myConnection.ServerType)
     myConnection.Close()
 End Sub

[C#]
public void CreateIfxConnection()
 {
    string myConnectString = "DATABASE=SAMPLE;";
    IfxConnection myConnection = new IfxConnection(myConnectString);
    myConnection.Open();
    MessageBox.Show("ServerType: " + myConnection.ServerType);
    myConnection.Close();
 }
```

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.ServerVersion Property:**

Gets a string containing the version of the server to which the client is connected.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public ReadOnly Property ServerVersion As String
[C#]
public string ServerVersion {get;}
[C++]
public: __property String* get_ServerVersion();
[JScript]
public function get ServerVersion() : String;
```

### Property value

The version of the connected server. The default value is an empty string ("") until the connection is opened.

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidOperationException | The connection is closed. |

### Remarks

The ServerVersion property takes the form '##.##.####,' where the first two digits are the major version, the next two digits are the minor version, and the last four digits are the release version.

### Example

[Visual Basic, C#] The following example creates a IfxConnection and displays the ServerVersion.

```
[Visual Basic]
Public Sub CreateIfxConnection()
     Dim myConnectString As String = "DATABASE=SAMPLE;"
     Dim myConnection As New IfxConnection(myConnectString)
     myConnection.Open()
     MessageBox.Show("ServerVersion: " + myConnection.ServerVersion.ToString())
     myConnection.Close()
 End Sub

[C#]
public void CreateIfxConnection()
 {
    string myConnectString = "DATABASE=SAMPLE;";
    IfxConnection myConnection = new IfxConnection(myConnectString);
    myConnection.Open();
    MessageBox.Show("ServerVersion: " + myConnection.ServerVersion.ToString());
    myConnection.Close();
 }
```

### Reference

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

## IfxConnection.State Property:

Gets the current state of the connection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property State As ConnectionState  Implements _
   IDbConnection.State
[C#]
public ConnectionState State {get;}
[C++]
public: __property ConnectionState get_State();
[JScript]
public function get State() : ConnectionState;
```

**Implements:**
IDbConnection.State

**Property value**

A bitwise combination of the System.Data.ConnectionState values. The default is Closed.

**Remarks**

The allowed state changes are:

- From Closed to Open, using the Open method of the IfxConnection object.
- From Open to Closed, using either the Close method or the Dispose method of the IfxConnection object.

**Note:** Calling the State property on an open connection increases application overhead because each such call results in an explicit low-level call to determine if the connection is still valid.

**Example**

[Visual Basic, C#] The following example creates an instance of a derived class, IfxConnection, sets its ConnectionString, and displays its State.

```
[Visual Basic]
Public Sub createIfxConnection()
    Dim myConnection As New IfxConnection()
    myConnection.ConnectionString = _
       "DATABASE=SAMPLE;"
    myConnection.Open()
    MessageBox.Show("Connection State: " + myConnection.State.ToString())
    myConnection.Close()
End Sub

[C#]
public void createIfxConnection()
{
   IfxConnection myConnection = new IfxConnection();
   myConnection.ConnectionString = "DATABASE=SAMPLE;";
   myConnection.Open();
   MessageBox.Show("Connection State: " + myConnection.State.ToString());
   myConnection.Close();
}
```

**Reference**

"IfxConnection Class" on page 5-157
"IfxConnection Members" on page 5-159
"IBM.Data.Informix Namespace" on page 5-1

## IfxConnection Events

The events of the IfxConnection class are listed here. For a complete list of IfxConnection class members, see the IfxConnection Members topic.

### Public Events

| Event | Description |
|---|---|
| Disposed (inherited from Component) | Adds an event handler to listen to the Disposed event on the component. |
| InfoMessage | Occurs when the IBM Data Server Provider for .NET sends a warning or an informational message. |
| StateChange | Occurs when the state of the connection changes. |

**Reference**

"IfxConnection Class" on page 5-157
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.InfoMessage Event:**

Occurs when the IBM Data Server Provider for .NET sends a warning or an informational message.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Event InfoMessage As IfxInfoMessageEventHandler

[C#]
public event IfxInfoMessageEventHandler
 InfoMessage;
[C++]
public: __event IfxInfoMessageEventHandler
* InfoMessage;
```

[JScript] In JScript(R), you can handle the events defined by a class, but you cannot declare new events.

**Event data**

| | |
|---|---|
| Errors | Gets the collection of warnings sent from the database. |

**Remarks**

Clients that want to process warnings or informational messages sent by the server should create a IfxInfoMessageEventHandler delegate to listen to this event.

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnection.StateChange Event:**

Occurs when the state of the connection changes.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Event StateChange As StateChangeEventHandler
[C#]
public event StateChangeEventHandler StateChange;
[C++]
public: __event StateChangeEventHandler* StateChange;
```

[JScript] In JScript(R), you can handle the events defined by a class, but you cannot declare new events.

*Table 5-17. Event Data*

| Item | Description |
|---|---|
| CurrentState | Gets the new state of the connection. The connection object will be in the new state already when the event is fired. |
| OriginalState | Gets the original state of the connection. |

**Remarks**

The StateChange event is raised immediately after the State changes from Closed to Opened, or from Opened to Closed.

If an event handler throws an exception from within the StateChange event, the exception propagates to the caller of the Open or Close methods.

**Reference**

"IfxConnection Class" on page 5-157

"IfxConnection Members" on page 5-159

"IBM.Data.Informix Namespace" on page 5-1

## IfxConnectionStringBuilder Class

Represents a simple approach for generating valid connection strings for use with the IfxConnectionString class.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
System.Data.Common.DbConnectionStringBuilder
IBM.Data.Informix.IfxConnectionStringBuilder

## Syntax

```
[Visual Basic]
Public NotInheritable Class IfxConnectionStringBuilder
 Inherits DbConnectionStringBuilder
[C#]
public sealed class IfxConnectionStringBuilder : DbConnectionStringBuilder
[C++]
public ref class IfxConnectionStringBuilder sealed :
  public DbConnectionStringBuilder
[JScript]
public final class IfxConnectionStringBuilder extends DbConnectionStringBuilder
```

## Example

[C#] The following example shows how to use a IfxConnectionStringBuilder instance to generate a valid IfxConnectionString.

**Note:** User ID and password information are included in this example for demonstration purposes. You should not hard-code this information in your applications.

```
[C#]
   IfxConnectionStringBuilder connStringBld = new IfxConnectionStringBuilder();
   connStringBld.Database = "SAMPLE";
   connStringBld.UserID = "Jack";
   connStringBld.Password = "BlueJays";
   connStringBld.Server = "jacksserver:db2c_DB2";

   IfxConnection conn = new IfxConnection(connStringBld.ConnectionString);
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnectionStringBuilder Members" on page 5-216

"IBM.Data.Informix Namespace" on page 5-1

## IfxConnectionStringBuilder Members

Represents a set of methods and properties that provide generic and IBM Data Server Provider for .NET-specific approaches to generating valid connection strings. The following tables list the members exposed by the IfxConnectionStringBuilder class.

### Public Constructors

| Constructor | Description |
|---|---|
| IfxConnectionStringBuilder() | Initializes a new instance of the IfxConnectionStringBuilder class. |
| IfxConnectionStringBuilder(string) | Initializes a new instance of the IfxConnectionStringBuilder class, using an existing ConnectionString. |

### Public Properties

| Property | Description |
|---|---|
| Authentication | Gets or sets the value of the authentication keyword. |
| CLISchema | Gets or sets a value indicating the schema to be used. If you set this property, the system catalog tables are not queried to generate the result sets. The static tables that you create using the **ifxocat** tool are queried instead. |
| ClientUserID | Gets or sets a client user ID that is sent to a database. |
| ClientWorkstationName | Gets or sets a client workstation name that is sent to a database. |
| Connect_Timeout | Gets or sets the number of seconds to wait before a connection attempt times out. |
| ConnectionLifeTime | Gets or sets the number of seconds that a connection can remain idle in the connection pool. |
| ConnectionReset | Gets or sets a value that determines whether a connection is placed in the connection pool when the connection is closed. |
| ConnectionString (inherited from DbConnectionStringBuilder) | Gets or sets the connection string associated with the DbConnectionStringBuilder. |
| Count (inherited from DbConnectionStringBuilder) | Gets the current number of keys that are in the ConnectionString property. |

| Property | Description |
| --- | --- |
|  CurrentSchema | Gets or sets the value of the schema name that is used for all unqualified SQL objects that are used in the connection. |
|  Database | Gets or sets the value of the database to connect to. |
|  Enlist | Gets or sets the value indicating whether automatic enlistment in the Distributed Transaction Coordinator (DTC) is enabled. |
|  IsFixedSize | Gets a value indicating whether the IfxConnectionStringBuilder has a fixed size. |
|  IsolationLevel | Gets or sets a value of the isolation level for the connection. |
|  IsReadOnly (inherited from DbConnectionStringBuilder) | Gets a value that indicates whether the DbConnectionStringBuilder is read only. |
|  Keys | Gets an ICollection that contains the keys in the IfxConnectionStringBuilder. |
|  MapDate | Gets or sets a value indicating whether DATE types are reported as a different data type. |
|  MapTime | Gets or sets a value indicating whether TIME types are reported as a different data type. |
|  MapTimestamp | Gets or sets a value indicating whether TIMESTAMP types are reported as a different data type. |
|  MaxPoolSize | Gets or sets a value representing the maximum connection pool size. |
|  MinPoolSize | Gets or sets a value representing the minimum connection pool size. |
|  Password | Gets or sets a value representing the user password. |

| Property | Description |
|---|---|
| PersistSecurityInfo | Gets or sets a value indicating whether security-sensitive information, such as a password, can be returned as part of a connection string after the connection has been opened or if the connection has ever been open. |
| Pooling | Gets or sets a value indicating whether connection pooling is turned on. |
| QueryTimeout | Gets or sets a value of the QueryTimeout keyword. |
| SchemaList | Gets or sets a value of the SchemaList keyword. |
| Server | Gets or sets a value specifying the host and port number to connect to. |
| SkipSynonymProcessing | Specifies whether IBM Data Server Provider for .NET sends a connection string to the DbPermission.Add method, without passing through the DB2ConnectionStringBuilder class. |
| StatementConcentrator | Gets or sets the value determining whether or not to override server configurations for statement concentrator literals. |
| SysSchema | Gets or sets a value of the SysSchema keyword. |
| TableType | Gets or sets a value of the TableType keyword. |
| UserID | Gets or sets a value representing the user name. |
| Values | Gets an ICollection that contains the values in the IfxConnectionStringBuilder. |

## Public Methods

| Method | Description |
|---|---|
| Add (inherited from DbConnectionStringBuilder) | Adds an entry with the specified key and value into the DbConnectionStringBuilder. |
| Clear | Clears the contents of the IfxConnectionStringBuilder instance. |

| Method | Description |
|---|---|
|  ContainsKey | Determines whether the IfxConnectionStringBuilder contains the specified key. |
|  Equals (inherited from Object) | Determines whether two Object instances are equal. |
|  EquivalentTo (inherited from DbConnectionStringBuilder) | Compares the connection information in this DbConnectionStringBuilder object with the connection information in the supplied object. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures like a hash table. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  Remove | Removes the entry with the specified key from the IfxConnectionStringBuilder. |
|  ShouldSerialize | Indicates whether the specified key exists in this IfxConnectionStringBuilder instance. |
|  ToString (inherited from DbConnectionStringBuilder) | Returns the connection string associated with this DbConnectionStringBuilder. |
|  TryGetValue | Retrieves a value from the IfxConnectionStringBuilder corresponding to the specified key. |

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

## IfxConnectionStringBuilder Constructors

Initializes a new instance of the IfxConnectionStringBuilder class.

### Public Constructors

| Constructor | Description |
|---|---|
|  IfxConnectionStringBuilder() | Initializes a new instance of the IfxConnectionStringBuilder class. |
|  IfxConnectionStringBuilder(string) | Initializes a new instance of the IfxConnectionStringBuilder class, using an existing ConnectionString. |

**Reference**

"IfxConnectionStringBuilder Members" on page 5-216
"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.IfxConnectionStringBuilder() Constructor:**

Initializes a new instance of the IfxConnectionStringBuilder class.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New()
[C#]
public IfxConnectionStringBuilder();
[C++]
public: IfxConnectionStringBuilder();
[JScript]
public function IfxConnectionStringBuilder();
```

**Remarks**

**Example**

[C#] The following example shows how to use a IfxConnectionStringBuilder instance to generate a valid IfxConnectionString.

**Note:** User ID and password information are included in this example for demonstration purposes. You should not hard-code this information in your applications.

```
[C#]
   IfxConnectionStringBuilder connStringBld = new IfxConnectionStringBuilder();
   connStringBld.Database = "SAMPLE";
   connStringBld.UserID = "Jack";
   connStringBld.Password = "BlueJays";
   connStringBld.Server = "jacksserver:db2c_DB2";

   IfxConnection conn = new IfxConnection(connStringBld.ConnectionString);
```

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.IfxConnectionStringBuilder(string) Constructor:**

Initializes a new instance of the IfxConnectionStringBuilder class, using an existing ConnectionString.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New(
   ByVal connectionString As String
)
[C#]
```

```
public IfxConnectionStringBuilder(
   string connectionString
);
[C++]
public: IfxConnectionStringBuilder(
   String* connectionString
);
[JScript]
public function IfxConnectionStringBuilder(
   connectionString : String
);
```

**Parameters**

**connectionString**
A connection string to be inserted into the IfxConnectionStringBuilder.

**Exceptions**

| Exception type | Condition |
|---|---|
| KeyNotFoundException | Invalid key name within the connection string. |
| FormatException | Invalid value within the connection string (for example, a string value was assigned when a numeric value was expected). |

**Remarks**

With this constructor, you can populate an instance of IfxConnectionStringBuilder with an existing connection string. This constructor will throw exceptions if invalid connection string keys or values are entered.

**Example**

[C#] The following example shows how to use a IfxConnectionStringBuilder instance to generate a valid IfxConnectionString, using an existing connection string.

**Note:** User ID and password information are included in this example for demonstration purposes. You should not hard-code this information in your applications.

```
[C#]
   String connString = "Database=SAMPLE;
   User ID=Jack;
   pwd=BlueJays;
   Server=jacksserver:db2c_DB2";
   IfxConnectionStringBuilder connStringBld = new
      IfxConnectionStringBuilder(connString);
```

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

## IfxConnectionStringBuilder Methods

The methods of the IfxConnectionStringBuilder class are listed here.

### Public Methods

| Method | Description |
|---|---|
| Add (inherited from DbConnectionStringBuilder) | Adds an entry with the specified key and value into the DbConnectionStringBuilder. |
| Clear | Clears the contents of the IfxConnectionStringBuilder instance. |
| ContainsKey | Determines whether the IfxConnectionStringBuilder contains the specified key. |
| Equals (inherited from Object) | Determines whether two Object instances are equal. |
| EquivalentTo (inherited from DbConnectionStringBuilder) | Compares the connection information in this DbConnectionStringBuilder object with the connection information in the supplied object. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures like a hash table. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| Remove | Removes the entry with the specified key from the IfxConnectionStringBuilder. |
| ShouldSerialize | Indicates whether the specified key exists in this IfxConnectionStringBuilder instance. |
| ToString (inherited from DbConnectionStringBuilder) | Returns the connection string associated with this DbConnectionStringBuilder. |
| TryGetValue | Retrieves a value from the IfxConnectionStringBuilder corresponding to the specified key. |

**Reference**

"IfxConnectionStringBuilder Members" on page 5-216
"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.Clear Method:**

Clears the contents of the IfxConnectionStringBuilder instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub Clear
[C#]
public void Close ()
[C++]
public:
void Close ()
[JScript]
public function Close ()
```

**Remarks**

This method clears all the connection string parameter key/value pairs, and sets the ConnectionString property to an empty string.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.ContainsKey Method:**

Determines whether the IfxConnectionStringBuilder contains the specified key.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Parameters**

**keyword**
The name of the key to find.

**Return value**

`true` if the specified key exists in the connection string; otherwise `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.Remove Method:**

Removes the entry with the specified key from the IfxConnectionStringBuilder.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Parameters**

**keyword**
The name of the key to remove.

**Return value**

`true` if the key was removed successfully; otherwise `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.ShouldSerialize Method:**

Indicates whether the specified key exists in this IfxConnectionStringBuilder instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Parameters**

**keyword**
The name of the key.

**Return value**

`true` if the specified key exists in the connection string; otherwise `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.TryGetValue Method:**

Retrieves a value from the IfxConnectionStringBuilder corresponding to the specified key.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Parameters**

**keyword**
The name of the key to use.

**value** The value of the specified key.

**Return value**

`true` if the key value pair exists in the connection string; otherwise `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

## IfxConnectionStringBuilder Properties

The properties of the IfxConnectionStringBuilder class are listed here.

### Public Properties

| Property | Description |
|---|---|
|  Authentication | Gets or sets the value of the authentication keyword. |
|  CLISchema | Gets or sets a value indicating the schema to be used. If you set this property, the system catalog tables are not queried to generate the result sets. The static tables that you create using the **ifxocat** tool are queried instead. |
|  ClientUserID | Gets or sets a client user ID that is sent to a database. |
|  ClientWorkstationName | Gets or sets a client workstation name that is sent to a database. |
|  Connect_Timeout | Gets or sets the number of seconds to wait before a connection attempt times out. |
|  ConnectionLifeTime | Gets or sets the number of seconds that a connection can remain idle in the connection pool. |
|  ConnectionReset | Gets or sets a value that determines whether a connection is placed in the connection pool when the connection is closed. |

| Property | Description |
| --- | --- |
|  ConnectionString (inherited from DbConnectionStringBuilder) | Gets or sets the connection string associated with the DbConnectionStringBuilder. |
|  Count (inherited from DbConnectionStringBuilder) | Gets the current number of keys that are in the ConnectionString property. |
|  CurrentSchema | Gets or sets the value of the schema name that is used for all unqualified SQL objects that are used in the connection. |
|  Database | Gets or sets the value of the database to connect to. |
|  Enlist | Gets or sets the value indicating whether automatic enlistment in the Distributed Transaction Coordinator (DTC) is enabled. |
|  IsFixedSize | Gets a value indicating whether the IfxConnectionStringBuilder has a fixed size. |
|  IsolationLevel | Gets or sets a value of the isolation level for the connection. |
|  IsReadOnly (inherited from DbConnectionStringBuilder) | Gets a value that indicates whether the DbConnectionStringBuilder is read only. |
|  Keys | Gets an ICollection that contains the keys in the IfxConnectionStringBuilder. |
|  MapDate | Gets or sets a value indicating whether DATE types are reported as a different data type. |
|  MapTime | Gets or sets a value indicating whether TIME types are reported as a different data type. |
|  MapTimestamp | Gets or sets a value indicating whether TIMESTAMP types are reported as a different data type. |
|  MaxPoolSize | Gets or sets a value representing the maximum connection pool size. |
|  MinPoolSize | Gets or sets a value representing the minimum connection pool size. |

| Property | Description |
|---|---|
| Password | Gets or sets a value representing the user password. |
| PersistSecurityInfo | Gets or sets a value indicating whether security-sensitive information, such as a password, can be returned as part of a connection string after the connection has been opened or if the connection has ever been open. |
| Pooling | Gets or sets a value indicating whether connection pooling is turned on. |
| QueryTimeout | Gets or sets a value of the QueryTimeout keyword. |
| SchemaList | Gets or sets a value of the SchemaList keyword. |
| Server | Gets or sets a value specifying the host and port number to connect to. |
| SkipSynonymProcessing | Specifies whether IBM Data Server Provider for .NET sends a connection string to the DbPermission.Add method, without passing through the DB2ConnectionStringBuilder class. |
| StatementConcentrator | Gets or sets the value determining whether or not to override server configurations for statement concentrator literals. |
| SysSchema | Gets or sets a value of the SysSchema keyword. |
| TableType | Gets or sets a value of the TableType keyword. |
| UserID | Gets or sets a value representing the user name. |
| Values | Gets an ICollection that contains the values in the IfxConnectionStringBuilder. |

**Reference**

"IfxConnectionStringBuilder Members" on page 5-216
"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.Authentication Property:**

Gets or sets the type of authentication to be used with file DSN or DSN-less connectivity.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Authentication As String
[C#]
public string Authentication {get; set;}
[C++]
public: __property String* get_Authentication();
public: __property void set_Authentication(String*);
[JScript]
public function get Authentication() : String;
public function set Authentication(String);
```

**Property value**

A string representing the type of authentication to be used. The default value is "SERVER".

**Remarks**

Valid values are:

- SERVER
- SERVER_ENCRYPT
- DATA_ENCRYPT
- KERBEROS
- GSSPLUGIN

When you set this option, you must also set the following options:

- Database;
- Protocol.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.CLISchema Property:**

Gets or sets the value indicating the schema to be used. When this keyword is set, the system catalog tables will not be queried to generate the result sets. The static tables created using the **ifxocat** tool will be queried instead.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property CLISchema As String
[C#]
public string CLISchema {get; set;}
[C++]
public: __property String* get_CLISchema();
public: __property void set_CLISchema(String*);
[JScript]
public function get CLISchema() : String;
public function set CLISchema(String);
```

**Property value**

A string representing the schema to be used with the **db2ocat** tables. By default, this value is not set.

**Note:** When connecting to a DB2 for z/OS database this keyword can be ignored or treated as a synonym for the SysSchema property.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.Connect_Timeout Property:**

Gets or sets the number of seconds for a connection attempt to timeout.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Connect_Timeout As int
[C#]
public int Connect_Timeout {get; set;}
[C++]
public: __property int* get_Connect_Timeout();
public: __property void set_Connect_Timeout(int*);
[JScript]
public function get Connect_Timeout() : int;
public function set Connect_Timeout(int);
```

**Property value**

An integer, greater than or equal 0 and less than or equal to 32767, that represents how many seconds the connection will wait before timing out. By default, the connection will wait indefinitely.

**Remarks**

If Connect_Timeout is set and client reroute is enabled, a connection will be attempted only once to the original server and once to the alternate server. Since the Connect_Timeout value is used when attempting to connect to each server, the maximum waiting time will be approximately double the Connect_Timeout value specified. This also applies when Sysplex exploitation is enabled, the maximum waiting time will be approximately equal to the number of Sysplex members, times the amount of time specified by the Connect_Timeout keyword.

Connect_Timeout only applies to the TCPIP protocol and is not supported for connections to databases cataloged on a SOCKS-enabled TCP/IP node.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.ConnectionLifeTime Property:**

Gets or sets the number of seconds that the connection can remain idle in the connection pool.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ConnectionLifeTime As int
[C#]
public int ConnectionLifeTime {get; set;}
[C++]
public: __property int* get_ConnectionLifeTime();
public: __property void set_ConnectionLifeTime(int*);
[JScript]
public function get ConnectionLifeTime() : int;
public function set ConnectionLifeTime(int);
```

**Property value**

An integer specifying the length of time a connection may remain idle in the connection pool.

**Note:**

- The default value is 60 seconds.
- Setting the property value to any positive integer specifies how long the connection will remain idle in the connection pool.
- A value of -1 means that the connection may remain indefinitely.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.ConnectionReset Property:**

Gets or sets a value that determines if the connection will be placed in the connection pool upon being closed.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ConnectionReset As Boolean
[C#]
public bool ConnectionReset {get; set;}
[C++]
public: __property bool* get_ConnectionReset();
public: __property void set_ConnectionReset(bool*);
[JScript]
public function get ConnectionReset() : Boolean;
public function set ConnectionReset(Boolean);
```

**Property value**

`true` if connections obtained by the IfxConnection.Open method should not be returned to the connection pool; otherwise `false`.

The default value of this property is `false`, the connection will be returned to the connection pool.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.ConnStrPwdEncrypt Property:**

Gets or sets the value indicating whether the connection string password is encrypted.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ConnStrPwdEncrypt As bool
[C#]
public bool ConnStrPwdEncrypt {get; set;}
[C++]
public: __property bool* get_ConnStrPwdEncrypt();
public: __property void set_ConnStrPwdEncrypt(bool*);
[JScript]
public function get ConnStrPwdEncrypt() : boolean;
public function set IfxConnectionStringBuilder(String);
```

**Property value**

A boolean value indicating whether the connection string password is encrypted.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.ConvertToLong Property:**

Gets or sets the value of the ConvertToLong keyword.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ConvertToLong As int
[C#]
public int ConvertToLong {get; set;}
[C++]
public: __property int* get_ConvertToLong();
public: __property void set_ConvertToLong(int*);
[JScript]
public function get ConvertToLong() : int;
public function set ConvertToLong(int);
```

**Property value**

**Version information**

**.NET Framework version**
Supported in: 2.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.CurrentSchema Property:**

Gets or sets the value of the schema name used for all unqualified SQL objects used in the connection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property CurrentSchema As String
[C#]
public string CurrentSchema {get; set;}
[C++]
public: __property String* get_CurrentSchema();
public: __property void set_CurrentSchema(String*);
[JScript]
public function get CurrentSchema() : String;
public function set CurrentSchema(String);
```

**Property value**

A string representing the default user ID to use as the owner for unqualified SQL objects.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.Database Property:**

Gets or sets the value of the database to connect to.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Database As String
[C#]
public string Database {get; set;}
[C++]
public: __property String* get_Database();
public: __property void set_Database(String*);
[JScript]
public function get Database() : String;
public function set Database(String);
```

**Property value**

A string value representing the database to connect to.

**Remarks**

This value is not related to any database alias names specified on the client. This value must be set to the name of the database on the server.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.Enlist Property:**

Gets or sets the value indicating whether automatic enlistment to the Distributed Transaction Coordinator (DTC) is enabled.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Enlist As Boolean
[C#]
public bool Enlist {get; set;}
[C++]
public: __property bool* get_Enlist();
public: __property void set_Enlist(bool*);
[JScript]
public function get Enlist() : Boolean;
public function set Enlist(Boolean);
```

**Property value**

`true` if automatic enlistment into a global transaction in enabled; otherwise `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.FitHighPrecisionType Property:**

Gets or sets the type of precision to be used:

**Namespace:**
IBM.Data.

**Assembly:**
IBM.Data. (in IBM.Data..dll)

**Syntax**

```
[Visual Basic]
Public Property FitHighPrecisionType As String
[C#]
public string FitHighPrecisionType {get; set;}
[C++]
public: __property String* get_Authentication();
public: __property void set_FitHighPrecisionType(String*);
[JScript]
public function get Authentication(): String;
public function set FitHighPrecisionType(String);
```

**Property value**

FitHighPrecisionType supports the following set of string values:

WithTruncate - Results in returning a .NET system type after silently truncating the column value if needed.

AsString- Results in converting the column to a .NET string type.

ReturnException - Results in a truncation exception if the value does not fit in the .NET system type.

**Version information**

**.NET Framework version**
Supported in: 2.0 , 3.0, 3.5 and 4.0

**IfxConnectionStringBuilder.Graphic Property:**

Gets or sets the value of the Graphic keyword.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Graphic As int
[C#]
public int Graphic {get; set;}
[C++]
public: __property int* get_Graphic();
public: __property void set_Graphic(int*);
[JScript]
public function get Graphic() : int;
public function set Graphic(int);
```

**Property value**

An integer, greater than or equal to 0 and less than or equal to 3, that represents the Graphic column length.

Set the Graphic keyword as follows:

- 0 - SQL_GRAPHIC is not returned as a supported SQL data type, and the reported length of GRAPHIC columns equals the maximum number of DBCS characters in the column.

- 1 - SQL_GRAPHIC is returned as a supported SQL data type, and the reported length of GRAPHIC columns equals the maximum number of DBCS characters in the column.
- 2 - SQL_GRAPHIC is not returned as a supported SQL data type, and the reported length of GRAPHIC columns equals the maximum number of bytes in the column.
- 3 - SQL_GRAPHIC is returned as a supported SQL data type, and the reported length of GRAPHIC columns equals the maximum number of bytes in the column.

**Version information**

**.NET Framework version**
Supported in: 2.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.IsFixedSize Property:**

Gets a value indicating whether the IfxConnectionStringBuilder has a fixed size.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property IsFixedSize As Boolean
[C#]
public bool IsFixedSize {get;}
[C++]
public: __property bool* get_IsFixedSize();
[JScript]
public function get IsFixedSize() : Boolean;
```

**Property value**

`true` if the IfxConnectionStringBuilder object has a fixed size; otherwise `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.IsolationLevel Property:**

Gets or sets the value of the isolation level for the connection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property IsolationLevel As int
[C#]
public int IsolationLevel {get; set;}
[C++]
public: __property int* get_IsolationLevel();
public: __property void set_IsolationLevel(int*);
[JScript]
public function get IsolationLevel() : int;
public function set IsolationLevel(int);
```

**Property value**

A string representing the isolation level used. The default value is "SQL_TXN_READ_COMMITTED".

Sets the isolation level to:

- SQL_TXN_READ_UNCOMMITTED - Read Uncommitted (Uncommitted read)
- SQL_TXN_READ_COMMITTED - Read Committed (Cursor stability)
- SQL_TXN_REPEATABLE_READ - Repeatable Read (Read Stability)
- SQL_TXN_SERIALIZABLE - Serializable (Repeatable read)

**Remarks**

This property is only applicable if the default isolation level is used. If the application specifically sets the isolation level for a connection or statement handle, then this property will have no effect on that handle.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.Keys Property:**

Gets an ICollection that contains the keys in the IfxConnectionStringBuilder.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Property value**

An ICollection containing the keys in the IfxConnectionStringBuilder object

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

### IfxConnectionStringBuilder.MapDate Property:

Gets or sets the value indicating whether the DATE types should be reported as a different data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property MapDate As int
[C#]
public int MapDate {get; set;}
[C++]
public: __property int* get_MapDate();
public: __property void set_MapDate(int*);
[JScript]
public function get MapDate() : int;
public function set MapDate(int);
```

**Property value**

An integer (0, 1, or 2) representing the mapping mode. The MapDate keyword is used to indicate whether the DATE types should be reported as a different data type.

Set the MapDate keyword as follows:

- 0 - No mapping is performed. Report DATE as DATE. (Default)
- 1 - Report DATE as SQL_CHAR.
- 2 - Report DATE as SQL_WCHAR.

**Note:** If MapDate is set to 1 or 2, then the ODBCVER keyword is ignored for TIMESTAMP columns.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

### IfxConnectionStringBuilder.MapTime Property:

Gets or sets the value indicating whether the TIME types should be reported as a different data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property MapTime As int
[C#]
public int MapTime {get; set;}
[C++]
public: __property int* get_MapTime();
public: __property void set_MapTime(int*);
[JScript]
public function get MapTime() : int;
public function set MapTime(int);
```

**Property value**

An integer (0, 1, or 2) representing the mapping mode. The MapDate keyword is used to indicate whether the TIME types should be reported as a different data type.

Set the MapDate keyword as follows:

- 0 - No mapping is performed. Report TIME as TIME. (Default)
- 1 - Report TIME as SQL_CHAR.
- 2 - Report TIME as SQL_WCHAR.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.MapTimestamp Property:**

Gets or sets the value indicating whether the TIMESTAMP types should be reported as a different data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property MapTimestamp As int
[C#]
public int MapTimestamp {get; set;}
[C++]
public: __property int* get_MapTimestamp();
public: __property void set_MapTimestamp(int*);
[JScript]
public function get MapTimestamp() : int;
public function set MapTimestamp(int);
```

**Property value**

An integer (0, 1, or 2) representing the mapping mode. The MapDate keyword is used to indicate whether the TIMESTAMP types should be reported as a different data type.

Set the MapDate keyword as follows:

- 0 - No mapping is performed. Report TIMESTAMP as TIMESTAMP. (Default)
- 1 - Report TIMESTAMP as SQL_CHAR.
- 2 - Report TIMESTAMP as SQL_WCHAR.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.MaxPoolSize Property:**

Gets or sets the value representing the maximum connection pool size.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property MaxPoolSize As int
[C#]
public int MaxPoolSize {get; set;}
[C++]
public: __property int* get_MaxPoolSize();
public: __property void set_MaxPoolSize(int*);
[JScript]
public function get MaxPoolSize() : int;
public function set MaxPoolSize(int);
```

**Property value**

An integer representing the maximum number of connections in the connection pool.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.MinPoolSize Property:**

Gets or sets the value representing the minimum connection pool size.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property MinPoolSize As int
[C#]
public int MinPoolSize {get; set;}
[C++]
public: __property int* get_MinPoolSize();
public: __property void set_MinPoolSize(int*);
[JScript]
public function get MinPoolSize() : int;
public function set MinPoolSize(int);
```

**Property value**

An integer representing the minimum number of connections in the connection pool.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.Password Property:**

Gets or sets the value representing the user password.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Password As String
[C#]
public string Password {get; set;}
[C++]
public: __property String* get_Password();
public: __property void set_Password(String*);
[JScript]
public function get Password() : String;
public function set Password(String);
```

**Property value**

A string representing the password to be used by the connection.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.PersistSecurityInfo Property:**

Gets or sets the value indicating if security-sensitive information, such as password, can be returned as part of the connection string after the connection has been opened or if the connection has ever been in an opened state.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property PersistSecurityInfoProperty As Boolean
[C#]
public bool PersistSecurityInfoProperty {get; set;}
[C++]
public: __property bool* get_PersistSecurityInfoProperty();
public: __property void set_PersistSecurityInfoProperty(bool*);
[JScript]
public function get PersistSecurityInfoProperty() : Boolean;
public function set PersistSecurityInfoProperty(Boolean);
```

**Property value**

`true` if security-sensitive information is returned as part of the connection string; otherwise `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.Pooling Property:**

Gets or sets the value representing whether connection pooling is turned on.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Pooling As Boolean
[C#]
```

```
public bool Pooling {get; set;}
[C++]
public: __property bool* get_Pooling();
public: __property void set_Pooling(bool*);
[JScript]
public function get Pooling() : Boolean;
public function set Pooling(Boolean);
```

**Property value**

`true` if pooling is enabled; otherwise `false`.

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder Properties:**

The properties of the IfxConnectionStringBuilder class are listed here.

**Public Properties**

| Property | Description |
|---|---|
|  Authentication | Gets or sets the value of the authentication keyword. |
|  CLISchema | Gets or sets a value indicating the schema to be used. If you set this property, the system catalog tables are not queried to generate the result sets. The static tables that you create using the **ifxocat** tool are queried instead. |
|  ClientUserID | Gets or sets a client user ID that is sent to a database. |
|  ClientWorkstationName | Gets or sets a client workstation name that is sent to a database. |
|  Connect_Timeout | Gets or sets the number of seconds to wait before a connection attempt times out. |
|  ConnectionLifeTime | Gets or sets the number of seconds that a connection can remain idle in the connection pool. |
|  ConnectionReset | Gets or sets a value that determines whether a connection is placed in the connection pool when the connection is closed. |

| Property | Description |
| --- | --- |
| ConnectionString (inherited from DbConnectionStringBuilder) | Gets or sets the connection string associated with the DbConnectionStringBuilder. |
| Count (inherited from DbConnectionStringBuilder) | Gets the current number of keys that are in the ConnectionString property. |
| CurrentSchema | Gets or sets the value of the schema name that is used for all unqualified SQL objects that are used in the connection. |
| Database | Gets or sets the value of the database to connect to. |
| Enlist | Gets or sets the value indicating whether automatic enlistment in the Distributed Transaction Coordinator (DTC) is enabled. |
| IsFixedSize | Gets a value indicating whether the IfxConnectionStringBuilder has a fixed size. |
| IsolationLevel | Gets or sets a value of the isolation level for the connection. |
| IsReadOnly (inherited from DbConnectionStringBuilder) | Gets a value that indicates whether the DbConnectionStringBuilder is read only. |
| Keys | Gets an ICollection that contains the keys in the IfxConnectionStringBuilder. |
| MapDate | Gets or sets a value indicating whether DATE types are reported as a different data type. |
| MapTime | Gets or sets a value indicating whether TIME types are reported as a different data type. |
| MapTimestamp | Gets or sets a value indicating whether TIMESTAMP types are reported as a different data type. |
| MaxPoolSize | Gets or sets a value representing the maximum connection pool size. |
| MinPoolSize | Gets or sets a value representing the minimum connection pool size. |

| Property | Description |
|---|---|
| Password | Gets or sets a value representing the user password. |
| PersistSecurityInfo | Gets or sets a value indicating whether security-sensitive information, such as a password, can be returned as part of a connection string after the connection has been opened or if the connection has ever been open. |
| Pooling | Gets or sets a value indicating whether connection pooling is turned on. |
| QueryTimeout | Gets or sets a value of the QueryTimeout keyword. |
| SchemaList | Gets or sets a value of the SchemaList keyword. |
| Server | Gets or sets a value specifying the host and port number to connect to. |
| SkipSynonymProcessing | Specifies whether IBM Data Server Provider for .NET sends a connection string to the DbPermission.Add method, without passing through the DB2ConnectionStringBuilder class. |
| StatementConcentrator | Gets or sets the value determining whether or not to override server configurations for statement concentrator literals. |
| SysSchema | Gets or sets a value of the SysSchema keyword. |
| TableType | Gets or sets a value of the TableType keyword. |
| UserID | Gets or sets a value representing the user name. |
| Values | Gets an ICollection that contains the values in the IfxConnectionStringBuilder. |

**Reference**

"IfxConnectionStringBuilder Members" on page 5-216
"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.QueryTimeout Property:**

Gets or sets the value of the QueryTimeout keyword.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property QueryTimeout As int
[C#]
public int QueryTimeout {get; set;}
[C++]
public: __property int* get_QueryTimeout();
public: __property void set_QueryTimeout(int*);
[JScript]
public function get QueryTimeout() : int;
public function set QueryTimeout(int);
```

**Property value**

An integer, greater than or equal to 0, representing the interval between query timeout checks. The default value is 5 seconds.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.SchemaList Property:**

Gets or sets the value of the SchemaList keyword.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property SchemaList As String
[C#]
public string SchemaList {get; set;}
[C++]
public: __property String* get_SchemaList();
public: __property void set_SchemaList(String*);
[JScript]
public function get SchemaList() : String;
public function set SchemaList(String);
```

**Property value**

A string representing the list of schemas to use. Each schema name is case-sensitive, must be delimited with single quotes, and separated by commas.

**Remarks**

The maximum length of the string is 256 characters.

**Example**

```
SchemaList  = "'USER1','USER2','USER3'"
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.Server Property:**

Gets or sets the value specifying the host and port number to connect to.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Server As String
[C#]
public string Server {get; set;}
[C++]
public: __property String* get_Server();
public: __property void set_Server(String*);
[JScript]
public function get Server() : String;
public function set Server(String);
```

**Property value**

A string representing the host and port to connect to.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**DB2ConnectionStringBuilder.SkipSynonymProcessing Property:**

Specifies whether IBM Data Server Provider for .NET sends a connection string to the DbPermission.Add method, without passing through the DB2ConnectionStringBuilder class.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property SkipSynonymProcessing As Boolean
[C#]
public bool SkipSynonymProcessing {set;}
[C++]
public: __property bool set_SkipSynonymProcessing();
[JScript]
public function set SkipSynonymProcessing():Boolean
```

**Property values**

If you set the SkipSynonymProcessing property to `true`, IBM Data Server Provider for .NET sends the connection string to the DbPermission.Add method, thereby saving the performance cost of the string manipulation that is required for processing synonym keywords.

The default value of the property is `false`.

**Remarks**

If you specify an invalid value for the SkipSynonymProcessing property, an ArgumentException exception is thrown.

**Version information**

**.NET Framework version**
Supported in 2.0, 3.0, 3.5, and 4.0

**IfxConnectionStringBuilder.StatementConcentrator Property:**

Gets or sets the value determining whether or not to override server configurations for statement concentrator literals.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property StatementConcentrator As string
[C#]
public string StatementConcentrator {get; set;}
[C++]
public: __property string* get_StatementConcentrator();
public: __property void set_StatementConcentrator(string*);
[JScript]
public function get StatementConcentrator() : string;
public function set StatementConcentrator(string);
```

**Property value**

A string representing whether or not statement concentrator literals are enabled.

Set the property to:

- `Off` - statement concentrator literals are disabled.
- `Literals` - statement concentrator literals are enabled. Literals will be converted to parameters.

**Remarks**

If this property is not set for the connection then the default behavior for statement concentrator literals will be determined by the server-side configuration.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0 3.5, and 4.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.StaticLatch Property:**

Gets or sets the value of the StaticLatch keyword.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property StaticLatch As int
[C#]
public int StaticLatch {get; set;}
[C++]
public: __property int* get_StaticLatch();
public: __property void set_StaticLatch(int*);
[JScript]
public function get StaticLatch() : int;
public function set StaticLatch(int);
```

**Property value**

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.SysSchema Property:**

Gets or sets the value of the SysSchema keyword.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property SysSchema As String
[C#]
public string SysSchema {get; set;}
[C++]
public: __property String* get_SysSchema();
public: __property void set_SysSchema(String*);
[JScript]
public function get SysSchema() : String;
public function set SysSchema(String);
```

**Property value**

A string representing an alternative schema to be searched.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

**IfxConnectionStringBuilder.TableType Property:**

Gets or sets the value of the TableType keyword.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property TableType As String
[C#]
public string TableType {get; set;}
[C++]
public: __property String* get_TableType();
public: __property void set_TableType(String*);
[JScript]
public function get TableType() : String;
public function set TableType(String);
```

**Property value**

A string representing the list of table types.

Valid table types to include in the list are:

- TABLE,
- ALIAS,
- VIEW,
- INOPERATIVE VIEW,
- SYSTEM TABLE,
- SYNONYM.

Any number of the values can be specified. Each type must be delimited with single quotes, separated by commas, and in uppercase.

**Remarks**

**Example**

```
TableType = "'TABLE','VIEW'"
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

### IfxConnectionStringBuilder.UserID Property:

Gets or sets the value representing the user name.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual&amp;nbsp;Basic]
Public Property UserID As String
[C#]
public string UserID {get; set;}
[C++]
public: __property String* get_UserID();
public: __property void set_UserID(String*);
[JScript]
public function get UserID() : String;
public function set UserID(String);
```

**Property value**

A string representing the user name to be used for the connection.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214
"IBM.Data.Informix Namespace" on page 5-1

### IfxConnectionStringBuilder.Values Property:

Gets the ICollection that contains the values in the IfxConnectionStringBuilder.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Property value**

An ICollection containing the values in the IfxConnectionStringBuilder.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxConnectionStringBuilder Class" on page 5-214

"IBM.Data.Informix Namespace" on page 5-1

# IfxCursorType Enumeration

The cursor types that the IfxResultSet instance can use.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
<Serializable>
Public Enum IfxCursorType
[C#]
[Serializable]
public enum IfxCursorType
[C++]
[Serializable]
__value public enum IfxCursorType
[JScript]
public
   Serializable
enum IfxCursorType
```

## Members

| Member name | Description |
|---|---|
| ForwardOnly | The IfxResultSet to be created will use a forward-only, non-scrollable, insensitive, and non-updatable cursor. |
| Static | The IfxResultSet to be created will use a scrollable, insensitive, and non-updatable cursor. |
| Keyset | The IfxResultSet to be created will use a scrollable, sensitive, and updatable cursor. Delete holes are visible to cursor. |
| Dynamic | The IfxResultSet to be created will use a scrollable, sensitive, dynamic, and updatable cursor. Delete holes are not visible to the cursor. |

## Remarks

The following table represents the IfxResultSet option settings that are assigned for particular IfxCursorType values:

| IfxCursorType value | IfxResultSet.Scrollable | IfxResultSet.Updatable |
|---|---|---|
| Forward-only | False | True if select contains primary-key, serial, or row-id |
| Static | True | False |

When creating a IfxResultSet instance with the IfxCommand.ExecuteResultSet method, you can use the IfxCursorType enumeration to define the capabilities of the IfxResultSet instance.

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataAdapter Class

Represents a set of data commands and a connection to a database that are used to fill the DataSet and update the database.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Inheritance hierarchy

System.Object
  System.MarshalByRefObject
    System.ComponentModel.Component
      System.Data.Common.DataAdapter
        System.Data.Common.DbDataAdapter
          IBM.Data.Informix.IfxDataAdapter

### Syntax

```
[Visual Basic]
NotInheritable Public Class IfxDataAdapter
   Inherits DbDataAdapter
   Implements IDbDataAdapter
[C#]
public sealed class IfxDataAdapter : DbDataAdapter, IDbDataAdapter
[C++]
public __gc __sealed class IfxDataAdapter : public DbDataAdapter,
   IDbDataAdapter
[JScript]
public class IfxDataAdapter extends DbDataAdapter implements
   IDbDataAdapter
```

### Remarks

The IfxDataAdapter serves as a bridge between a DataSet and database for retrieving and saving data. The IfxDataAdapter provides this bridge by using Fill to load data from the database into the DataSet, and using Update to send changes made in the DataSet back to the database.

**Note:** When you call the Fill method on a database that does not have a primary key column, the IfxDataAdapter attempts to promote the unique constraint column to the primary key. In the process, the IfxDataAdapter marks the unique constraint as not nullable. This behavior works unless there is a null value in the unique constraint column. If there is a null value, the Fill method fails with a constraint violation. To avoid this situation, do not allow null values in the unique constraint column.

The IfxDataAdapter also includes the SelectCommand, InsertCommand, DeleteCommand, UpdateCommand, and TableMappings properties to facilitate loading and updating of data.

## Example

[Visual Basic, C#] The following example uses IfxCommand , IfxDataAdapter and IfxConnection to select records, and populate a DataSet with the selected rows. The filled DataSet is then returned. To accomplish this, the method is passed an initialized DataSet, with the SelectCommand set with the specified SQL SELECT statement and a connection to the database.

```
[Visual Basic]
Public Function SelectIfxSrvRows(dataSet As DataSet,
    connection As String, query As String) As DataSet
    Dim conn As New IfxConnection(connection)
    Dim adapter As New IfxDataAdapter()
    adapter.SelectCommand = new IfxCommand(query, conn)
    adapter.Fill(dataset)
    Return dataset
End Function
[C#]
public DataSet SelectIfxSrvRows(DataSet dataset,string connection,string query)
{
    IfxConnection conn = new IfxConnection(connection);
    IfxDataAdapter adapter = new IfxDataAdapter();
    adapter.SelectCommand = new IfxCommand(query, conn);
    adapter.Fill(dataset);
    return dataset;
}
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataAdapter Members"

"IBM.Data.Informix Namespace" on page 5-1

"IfxConnection Class" on page 5-157

"IfxCommand Class" on page 5-110

## IfxDataAdapter Members

IfxDataAdapter overview

## Public Constructors

| Name | Description |
|---|---|
|  IfxDataAdapter | Overloaded. Initializes a new instance of the IfxDataAdapter class. |

## Public Properties

| Name | Description |
|---|---|
|  AcceptChangesDuringFill (inherited from DataAdapter) | Gets or sets a value indicating whether AcceptChanges is called on a DataRow after it is added to the DataTable. |
|  Container (inherited from Component) | Gets the IContainer that contains the Component. |
|  ContinueUpdateOnError (inherited from DataAdapter) | Gets or sets a value that specifies whether to generate an exception, or the row in error when an error is encountered during a row update. |
|  DeleteCommand | Gets or sets an SQL statement or stored procedure used to delete records in the database. |
|  InsertCommand | Gets or sets an SQL statement or stored procedure used to insert new records into the database. |
|  MissingMappingAction (inherited from DataAdapter) | Determines the action to take when incoming data does not have a matching table or column. |
|  MissingSchemaAction (inherited from DataAdapter) | Determines the action to take when existing DataSet schema does not match incoming data. |
|  SelectCommand | Gets or sets an SQL statement or stored procedure used to select records in the database. |
|  Site (inherited from Component) | Gets or sets the ISite of the Component. |
|  TableMappings (inherited from DataAdapter) | Gets a collection that provides the master mapping between a source table and a DataTable. |
|  UpdateBatchSize | Overloaded. Gets or sets the number of commands in a batch to be sent to the database server for execution. |
|  UpdateCommand | Gets or sets an SQL statement or stored procedure used to update records in the database. |

## Public Methods

| Name | Description |
|---|---|
| CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
| Dispose (inherited from Component) | Overloaded. Releases the resources used by the Component. |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| Fill (inherited from DbDataAdapter) | Overloaded. Overridden. Adds or refreshes rows in the DataSet to match those in the database. |
| FillSchema (inherited from DbDataAdapter) | Overloaded. Overridden. Adds a DataTable to a DataSet and configures the schema to match that in the database. |
| GetFillParameters (inherited from DbDataAdapter) | Overridden. Gets the parameters set by the user when executing an SQL SELECT statement. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |
| Update (inherited from DbDataAdapter) | Overloaded. Calls the respective INSERT, UPDATE, or DELETE statements for each inserted, updated, or deleted row in the DataSet. |

## Public Events

| Name | Description |
|---|---|
| Disposed (inherited from Component) | Overloaded. Adds an event handler to listen to the Disposed event on the component. |
| FillError (inherited from DbDataAdapter) | Returned when an error occurs during a fill operation. |
| RowUpdated | Occurs during an update operation after a command is executed against the database. |
| RowUpdating | Occurs during Update before a command is executed against the database. |

## Protected Properties

| Name | Description |
|---|---|
|  DesignMode (inherited from Component) | Gets a value that indicates whether the Component is currently in design mode. |
|  Events (inherited from Component) | Gets the list of event handlers that are attached to this Component. |

## Protected Methods

| Name | Description |
|---|---|
|  AddToBatch (inherited from DbDataAdapter) | Adds a IDbCommand to the current batch. |
|  ClearBatch (inherited from DbDataAdapter) | Removes all IDbCommand objects from the batch. |
|  CloneInternals (inherited from DataAdapter) | Creates a copy of this instance of DataAdapter. |
|  CreateRowUpdatedEvent | Overridden. Initializes a new instance of the RowUpdatedEventArgs class, regardless of whether the update is successful. |
|  CreateRowUpdatingEvent | Overridden. Initializes a new instance of the RowUpdatingEventArgs class. |
|  CreateTableMappings (inherited from DataAdapter) | Creates a new DataTableMappingCollection. |
|  Dispose (inherited from DbDataAdapter) | Overloaded. Releases the resources used by the Component. |
|  ExecuteBatch (inherited from DbDataAdapter) | Executes the current batch. |
|  Fill (inherited from DbDataAdapter) | Overloaded. Overridden. Adds or refreshes rows in the DataSet to match those in the database. |
|  FillSchema (inherited from DbDataAdapter) | Overloaded. Overridden. Adds a DataTable to a DataSet and configures the schema to match that in the database. |
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |

| Name | Description |
|---|---|
| GetBatchedParameter (inherited from DbDataAdapter) | Returns a IDataParameter from one of the commands in the current batch. |
| GetService (inherited from Component) | Returns an object that represents a service provided by the Component or by its Container. |
| InitializeBatching (inherited from DbDataAdapter) | Initializes batching for the DbDataAdapter. |
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |
| OnFillError (inherited from DbDataAdapter) | Raises the FillError event. |
| OnRowUpdated | Overridden. Raises the RowUpdated event using a RowUpdatedEventArgs object. |
| OnRowUpdating | Overridden. Raises the RowUpdating event using a RowUpdatingEventArgs object, whether or not the update operation is successful. |
| ShouldSerializeTableMappings (inherited from DataAdapter) | Determines whether one or more DataTableMapping objects exist and they should be persisted. |
| TerminateBatching (inherited from DbDataAdapter) | Ends batching for the DbDataAdapter. |

**Reference**

"IfxDataAdapter Class" on page 5-253

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataAdapter Constructor

Initializes a new instance of the IfxDataAdapter class.

### Overload list

| Name | Description |
|---|---|
| New() | Initializes a new instance of the IfxDataAdapter class. |
| New(IfxCommand) | Initializes a new instance of the IfxDataAdapter class with the specified SQL SELECT statement. |
| New(String, IfxConnection) | Initializes a new instance of the IfxDataAdapter class with an SQL SELECT statement and a IfxConnection . |
| New(String, String) | Initializes a new instance of the IfxDataAdapter class with an SQL SELECT statement and a connection string. |

## Example

[Visual Basic, C#] The following example creates a IfxDataAdapter and sets some of its properties.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of the IfxDataAdapter constructor. For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub CreateIfxDataAdapter()
    Dim myIfxConnection As IfxConnection = New IfxConnection("DATABASE=SAMPLE;")
    Dim mySelectText as String = "SELECT EMPNO, LASTNAME FROM EMPLOYEE"
    Dim empDA As IfxDataAdapter = New IfxDataAdapter(mySelectText, myIfxConnection)
    empDA.MissingSchemaAction = MissingSchemaAction.AddWithKey

    Dim employeeDS As DataSet = New DataSet()
    empDA.Fill(employeeDS, "EMPLOYEE")

    empDA.InsertCommand =
      New IfxCommand("INSERT INTO EMPLOYEE (EMPNO, LASTNAME) " & _
      "VALUES (?, ?)", myIfxConnection)
    empDA.UpdateCommand =
      New IfxCommand("UPDATE EMPLOYEE SET EMPNO = ?, LASTNAME = ? " & _
      "WHERE EMPNO = ?", myIfxConnection)
    empDA.DeleteCommand =
      New IfxCommand("DELETE FROM EMPLOYEE
      WHERE EMPNO = ?", myIfxConnection)

    empDA.InsertCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO")
    empDA.InsertCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME")

    empDA.UpdateCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO")
    empDA.UpdateCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME")
    empDA.UpdateCommand.Parameters.Add("OLDEMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original

empDA.DeleteCommand.Parameters.Add("EMPNO", IfxType.Char, 5,
  "EMPNO").SourceVersion = DataRowVersion.Original
End Sub
[C#]
public static void CreateIfxDataAdapter()
{
    IfxConnection myIfxConnection = new IfxConnection("DATABASE=SAMPLE;");

    string mySelectText = "SELECT EMPNO, LASTNAME FROM EMPLOYEE";

    IfxDataAdapter empDA = new IfxDataAdapter(mySelectText, myIfxConnection );
    empDA.MissingSchemaAction = MissingSchemaAction.AddWithKey;

    DataSet employeeDS = new DataSet();
    empDA.Fill(employeeDS, "EMPLOYEE");

    empDA.InsertCommand = new IfxCommand("INSERT INTO EMPLOYEE
      (EMPNO, LASTNAME) " + "VALUES (?, ?)", myIfxConnection);
    empDA.UpdateCommand = new IfxCommand("UPDATE EMPLOYEE
      SET EMPNO = ?, LASTNAME = ? " +
      "WHERE EMPNO = ?", myIfxConnection);
    empDA.DeleteCommand = new IfxCommand("DELETE FROM EMPLOYEE
      WHERE EMPNO = ?", myIfxConnection);

    empDA.InsertCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO");
    empDA.InsertCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME");

    empDA.UpdateCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO");
    empDA.UpdateCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME");
```

```
    empDA.UpdateCommand.Parameters.Add("OLDEMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original;

    empDA.DeleteCommand.Parameters.Add("EMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original;
}
```

**Reference**

"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

### IfxDataAdapter Constructor ():

Initializes a new instance of the IfxDataAdapter class.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New()
[C#]
public IfxDataAdapter();
[C++]
public: IfxDataAdapter();
[JScript]
public function IfxDataAdapter();
```

**Remarks**

When you create an instance of IfxDataAdapter , the following read/write properties are set to their default values, as shown in the table.

| Properties | Default value |
|---|---|
| MissingMappingAction | MissingMappingAction.Passthrough |
| MissingSchemaAction | MissingSchemaAction.Add |

You can change the value of any of these properties through a separate call to the property.

**Example**

[Visual Basic, C#] The following example creates a IfxDataAdapter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxDataAdapter()
    Dim myIfxConnection As IfxConnection = New IfxConnection("DATABASE=SAMPLE;")
    Dim empDA As IfxDataAdapter = New IfxDataAdapter
    empDA.MissingSchemaAction = MissingSchemaAction.AddWithKey

    Dim employeeDS As DataSet = New DataSet
    empDA.Fill(employeeDS, "EMPLOYEE")

    empDA.SelectCommand = New IfxCommand(
      "SELECT EMPNO, LASTNAME FROM EMPLOYEE", myIfxConnection)
```

```
    empDA.InsertCommand = New IfxCommand(
      "INSERT INTO EMPLOYEE (EMPNO, LASTNAME) " & _
      "VALUES (?, ?)", myIfxConnection)
    empDA.UpdateCommand = New IfxCommand(
      "UPDATE EMPLOYEE SET EMPNO = ?, LASTNAME = ? " & _
      "WHERE EMPNO = ?", myIfxConnection)
    empDA.DeleteCommand = New IfxCommand(
      "DELETE FROM EMPLOYEE WHERE EMPNO = ?",
      myIfxConnection)

    empDA.InsertCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO")
    empDA.InsertCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME")

    empDA.UpdateCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO")
    empDA.UpdateCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME")
    empDA.UpdateCommand.Parameters.Add("OLDEMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original

    empDA.DeleteCommand.Parameters.Add("EMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original
End Sub

[C#]
public static void CreateIfxDataAdapter()
{
    IfxConnection myIfxConnection = new IfxConnection("DATABASE=SAMPLE;");
    IfxDataAdapter empDA = new IfxDataAdapter();
    empDA.MissingSchemaAction = MissingSchemaAction.AddWithKey;

    DataSet employeeDS = new DataSet();
    empDA.Fill(employeeDS, "EMPLOYEE");

    empDA.SelectCommand = new IfxCommand(
      "SELECT EMPNO, LASTNAME FROM EMPLOYEE", myIfxConnection);
    empDA.InsertCommand = new IfxCommand(
      "INSERT INTO EMPLOYEE (EMPNO, LASTNAME) " +
      "VALUES (?, ?)", myIfxConnection);
    empDA.UpdateCommand = new IfxCommand(
      "UPDATE EMPLOYEE SET EMPNO = ?, LASTNAME = ? " +
      "WHERE EMPNO = ?", myIfxConnection);
    empDA.DeleteCommand = new IfxCommand(
      "DELETE FROM EMPLOYEE WHERE EMPNO = ?", myIfxConnection);

    empDA.InsertCommand.Parameters.Add(
      "EMPNO", IfxType.Char, 6, "EMPNO");
    empDA.InsertCommand.Parameters.Add(
      "LASTNAME", IfxType.VarChar, 15, "LASTNAME");

    empDA.UpdateCommand.Parameters.Add(
      "EMPNO", IfxType.Char, 6, "EMPNO");
    empDA.UpdateCommand.Parameters.Add(
      "LASTNAME", IfxType.VarChar, 15, "LASTNAME");
    empDA.UpdateCommand.Parameters.Add(
      "OLDEMPNO", IfxType.Char, 6, "EMPNO").SourceVersion = DataRowVersion.Original;

    empDA.DeleteCommand.Parameters.Add(
      "EMPNO", IfxType.Char, 6, "EMPNO").SourceVersion = DataRowVersion.Original;
}
```

**Reference**


"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

"IfxDataAdapter Constructor" on page 5-258

### IfxDataAdapter Constructor (IfxCommand):

Initializes a new instance of the IfxDataAdapter class with the specified SQL SELECT statement.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New( _
   ByVal selectCommand As IfxCommand

)_
[C#]
public IfxDataAdapter(
   IfxCommand
 selectCommand
);
[C++]
public: IfxDataAdapter(
   IfxCommand
* selectCommand
);
[JScript]
public function IfxDataAdapter(
   selectCommand : IfxCommand

);
```

**Parameters**

**selectCommand**
A IfxCommand that is an SQL SELECT statement or stored procedure, and is set as the SelectCommand property of the IfxDataAdapter.

**Remarks**

This implementation of the IfxDataAdapter constructor sets the SelectCommand property to the value specified in the **selectCommand** parameter.

When you create an instance of IfxDataAdapter, the following read/write properties are set to their default values, as shown in the table.

| Properties | Initial value |
|---|---|
| MissingMappingAction | MissingMappingAction.Passthrough |
| MissingSchemaAction | MissingSchemaAction.Add |

You can change the value of any of these properties through a separate call to the property.

**Example**

[Visual Basic, C#] The following example creates a IfxDataAdapter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxDataAdapter()
    Dim myIfxConnection As IfxConnection = New IfxConnection("DATABASE=SAMPLE;")
    Dim myIfxCommand As IfxCommand =
      New IfxCommand("SELECT EMPNO, LASTNAME FROM EMPLOYEE");
    Dim custDA As IfxDataAdapter = New IfxDataAdapter(myIfxCommand)
    custDA.MissingSchemaAction = MissingSchemaAction.AddWithKey

    Dim custDS As DataSet = New DataSet
    custDA.Fill(custDS, "Customers")

    custDA.InsertCommand = New IfxCommand(
      "INSERT INTO EMPLOYEE (EMPNO, LASTNAME) " & _
      "VALUES (?, ?)", myIfxConnection)
    custDA.UpdateCommand = New IfxCommand(
      "UPDATE EMPLOYEE SET EMPNO = ?, LASTNAME = ? " & _
      "WHERE EMPNO = ?", myIfxConnection)
    custDA.DeleteCommand = New IfxCommand(
      "DELETE FROM EMPLOYEE WHERE EMPNO = ?", myIfxConnection)

    custDA.InsertCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO")
    custDA.InsertCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME")

    custDA.UpdateCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO")
    custDA.UpdateCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME")
    custDA.UpdateCommand.Parameters.Add(
      "OLDEMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original

    custDA.DeleteCommand.Parameters.Add(
      "EMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original
End Sub

[C#]
public static void CreateIfxDataAdapter()
{
    IfxConnection myIfxConnection = new IfxConnection("DATABASE=SAMPLE;");
    IfxCommand myIfxCommand =
      new IfxCommand("SELECT EMPNO, LASTNAME FROM EMPLOYEE");
    IfxDataAdapter custDA = new IfxDataAdapter(myIfxCommand);
    custDA.MissingSchemaAction = MissingSchemaAction.AddWithKey;

    DataSet custDS = new DataSet();
    custDA.Fill(employeeDS, "EMPLOYEE");

    custDA.InsertCommand = new IfxCommand(
      "INSERT INTO EMPLOYEE (EMPNO, LASTNAME) " +
      "VALUES (?, ?)", myIfxConnection);
    custDA.UpdateCommand = new IfxCommand(
      "UPDATE EMPLOYEE SET EMPNO = ?, LASTNAME = ? " +
      "WHERE EMPNO = ?", myIfxConnection);
    custDA.DeleteCommand = new IfxCommand(
      "DELETE FROM EMPLOYEE WHERE EMPNO = ?",
      myIfxConnection);

    custDA.InsertCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO");
    custDA.InsertCommand.Parameters.Add(
      "LASTNAME", IfxType.VarChar, 15, "LASTNAME");

    custDA.UpdateCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO");
    custDA.UpdateCommand.Parameters.Add(
      "LASTNAME", IfxType.VarChar, 15, "LASTNAME");
    custDA.UpdateCommand.Parameters.Add(
      "OLDEMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original;
```

```
    custDA.DeleteCommand.Parameters.Add(
      "EMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original;
}
```

**Reference**

"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

"IfxDataAdapter Constructor" on page 5-258

**IfxDataAdapter Constructor (String, IfxConnection):**

Initializes a new instance of the IfxDataAdapter class with an SQL SELECT statement and a IfxConnection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New( _
   ByVal selectCommandText As String, _
   ByVal selectConnection As IfxConnection _
)
[C#]
public IfxDataAdapter(
   string selectCommandText,
   IfxConnection
 selectConnection
);
[C++]
public: IfxDataAdapter(
   String* selectCommandText,
   IfxConnection* selectConnection
);
[JScript]
public function IfxDataAdapter(
   selectCommandText : String,
   selectConnection : IfxConnection
);
```

**Parameters**

**selectCommandText**
A string that is a SQL SELECT statement or stored procedure to be used by the SelectCommand property of the IfxDataAdapter.

**selectConnection**
A IfxConnection object that represents an open connection to a database.

**Remarks**

This implementation of the IfxDataAdapter can be useful in an application that must call the Fill method for two or more IfxDataAdapter objects.

**Example**

[Visual Basic, C#] The following example creates a IfxDataAdapter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxDataAdapter()
    Dim myIfxConnection As IfxConnection = New IfxConnection("DATABASE=SAMPLE;")
    Dim mySelectText As String = "SELECT EMPNO, LASTNAME FROM EMPLOYEE"
    Dim custDA As IfxDataAdapter = New IfxDataAdapter(mySelectText, myIfxConnection)
    custDA.MissingSchemaAction = MissingSchemaAction.AddWithKey

    Dim custDS As DataSet = New DataSet
    custDA.Fill(employeeDS, "EMPLOYEE")

    custDA.InsertCommand = New IfxCommand(
      "INSERT INTO EMPLOYEE (EMPNO, LASTNAME) " & _
      "VALUES (?, ?)", myIfxConnection)
    custDA.UpdateCommand = New IfxCommand(
      "UPDATE EMPLOYEE SET EMPNO = ?, LASTNAME = ? " & _
      "WHERE EMPNO = ?", myIfxConnection)
    custDA.DeleteCommand = New IfxCommand(
      "DELETE FROM EMPLOYEE WHERE EMPNO = ?",
      myIfxConnection)

    custDA.InsertCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO")
    custDA.InsertCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME")

    custDA.UpdateCommand.Parameters.Add(
      "EMPNO", IfxType.Char, 6, "EMPNO")
    custDA.UpdateCommand.Parameters.Add(
      "LASTNAME", IfxType.VarChar, 15, "LASTNAME")
    custDA.UpdateCommand.Parameters.Add(
      "OLDEMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original

    custDA.DeleteCommand.Parameters.Add(
      "EMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original
End Sub

[C#]
public static void CreateIfxDataAdapter()
{
    IfxConnection myIfxConnection = new IfxConnection("DATABASE=SAMPLE;");

    string mySelectText = "SELECT CustomerID, CompanyName FROM CUSTOMERS";

    IfxDataAdapter custDA = new IfxDataAdapter(mySelectText, myIfxConnection );
    custDA.MissingSchemaAction = MissingSchemaAction.AddWithKey;

    DataSet custDS = new DataSet();
    custDA.Fill(custDS, "EMPLOYEE");

    custDA.InsertCommand = new IfxCommand(
      "INSERT INTO EMPLOYEE (EMPNO, LASTNAME) " +
      "VALUES (?, ?)", myIfxConnection);
    custDA.UpdateCommand = new IfxCommand(
      "UPDATE EMPLOYEE SET EMPNO = ?, LASTNAME = ? " +
      "WHERE EMPNO = ?", myIfxConnection);
    custDA.DeleteCommand = new IfxCommand(
      "DELETE FROM EMPLOYEE WHERE EMPNO = ?",
      myIfxConnection);

    custDA.InsertCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO");
    custDA.InsertCommand.Parameters.Add(
      "LASTNAME", IfxType.VarChar, 15, "LASTNAME");
```

```
    custDA.UpdateCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO");
    custDA.UpdateCommand.Parameters.Add(
      "LASTNAME", IfxType.VarChar, 15, "LASTNAME");
    custDA.UpdateCommand.Parameters.Add("OLDEMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original;

    custDA.DeleteCommand.Parameters.Add("EMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original;
}
```

**Reference**

"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

"IfxDataAdapter Constructor" on page 5-258

### IfxDataAdapter Constructor (String, String):

Initializes a new instance of the IfxDataAdapter class with an SQL SELECT statement and a connection string.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New( _
   ByVal selectCommandText As String, _
   ByVal selectConnectionString As String _
)
[C#]
public IfxDataAdapter(
   string selectCommandText,
   string selectConnectionString
);
[C++]
public: IfxDataAdapter(
   String* selectCommandText,
   String* selectConnectionString
);
[JScript]
public function IfxDataAdapter(
   selectCommandText : String,
   selectConnectionString : String
);
```

**Parameters**

**selectCommandText**
A string that is a SQL SELECT statement or stored procedure to be used by the SelectCommand property of the IfxDataAdapter.

**selectConnectionString**
The connection string.

**Example**

[Visual Basic, C#] The following example creates a IfxDataAdapter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxDataAdapter()
    Dim myConnString As String = "DATABASE=SAMPLE;"
    Dim myIfxConnection As IfxConnection = New IfxConnection(myConnString)
    Dim mySelectText As String = "SELECT EMPNO, LASTNAME FROM EMPLOYEE"
    Dim empDA As IfxDataAdapter = New IfxDataAdapter(mySelectText, myIfxConnection)
    empDA.MissingSchemaAction = MissingSchemaAction.AddWithKey

    Dim employeeDS As DataSet = New DataSet()
    empDA.Fill(employeeDS, "EMPLOYEES")

    empDA.InsertCommand = New IfxCommand(
      "INSERT INTO EMPLOYEE (EMPNO, LASTNAME) " & _
      "VALUES (?, ?)", myIfxConnection)
    empDA.UpdateCommand = New IfxCommand(
      "UPDATE EMPLOYEE SET EMPNO = ?, LASTNAME = ? " & _
      "WHERE EMPNO = ?", myIfxConnection)
    empDA.DeleteCommand = New IfxCommand(
      "DELETE FROM EMPLOYEE WHERE EMPNO = ?",
      myIfxConnection)

    empDA.InsertCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO")
    empDA.InsertCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME")

    empDA.UpdateCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO")
    empDA.UpdateCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME")
    empDA.UpdateCommand.Parameters.Add("OLDEMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original

    empDA.DeleteCommand.Parameters.Add("EMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original
End Sub

[C#]
public static void CreateIfxDataAdapter()
{

    string myConnString = "DATABASE=SAMPLE;";
    IfxConnection myIfxConnection = new IfxConnection(myConnString);
    string mySelectText = "SELECT EMPNO, LASTNAME FROM EMPLOYEE";
    IfxDataAdapter empDA = new IfxDataAdapter(mySelectText, myIfxConnection);
    empDA.MissingSchemaAction = MissingSchemaAction.AddWithKey;

    DataSet employeeDS = new DataSet();
    empDA.Fill(employeeDS, "EMPLOYEES");

    empDA.InsertCommand = new IfxCommand(
      "INSERT INTO EMPLOYEE (EMPNO, LASTNAME) " +
      "VALUES (?, ?)", myIfxConnection);
    empDA.UpdateCommand = new IfxCommand(
      "UPDATE EMPLOYEE SET EMPNO = ?, LASTNAME = ? " +
      "WHERE EMPNO = ?", myIfxConnection);
    empDA.DeleteCommand = new IfxCommand(
      "DELETE FROM EMPLOYEE WHERE EMPNO = ?", myIfxConnection);

    empDA.InsertCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO");
    empDA.InsertCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME");

    empDA.UpdateCommand.Parameters.Add("EMPNO", IfxType.Char, 6, "EMPNO");
    empDA.UpdateCommand.Parameters.Add("LASTNAME", IfxType.VarChar, 15, "LASTNAME");
    empDA.UpdateCommand.Parameters.Add("OLDEMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original;
```

```
    empDA.DeleteCommand.Parameters.Add("EMPNO", IfxType.Char, 6,
      "EMPNO").SourceVersion = DataRowVersion.Original;
}
```

**Reference**

"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

"IfxDataAdapter Constructor" on page 5-258

## IfxDataAdapter Methods

The methods of the IfxDataAdapter class are listed here. For a complete list of IfxDataAdapter class members, see the IfxDataAdapter Members topic.

### Public Methods

| Name | Description |
|---|---|
| CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
| Dispose (inherited from Component) | Overloaded. Releases the resources used by the Component. |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| Fill (inherited from DbDataAdapter) | Overloaded. Overridden. Adds or refreshes rows in the DataSet to match those in the database. |
| FillSchema (inherited from DbDataAdapter) | Overloaded. Overridden. Adds a DataTable to a DataSet and configures the schema to match that in the database. |
| GetFillParameters (inherited from DbDataAdapter) | Overridden. Gets the parameters set by the user when executing an SQL SELECT statement. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |
| Update (inherited from DbDataAdapter) | Overloaded. Calls the respective INSERT, UPDATE, or DELETE statements for each inserted, updated, or deleted row in the DataSet. |

## Protected Methods

| Name | Description |
|---|---|
|  AddToBatch (inherited from DbDataAdapter) | Adds a IDbCommand to the current batch. |
|  ClearBatch (inherited from DbDataAdapter) | Removes all IDbCommand objects from the batch. |
|  CloneInternals (inherited from DataAdapter) | Creates a copy of this instance of DataAdapter. |
|  CreateRowUpdatedEvent | Overridden. Initializes a new instance of the RowUpdatedEventArgs class, regardless of whether the update is successful. |
|  CreateRowUpdatingEvent | Overridden. Initializes a new instance of the RowUpdatingEventArgs class. |
|  CreateTableMappings (inherited from DataAdapter) | Creates a new DataTableMappingCollection. |
|  Dispose (inherited from DbDataAdapter) | Overloaded. Releases the resources used by the Component. |
|  ExecuteBatch (inherited from DbDataAdapter) | Executes the current batch. |
|  Fill (inherited from DbDataAdapter) | Overloaded. Overridden. Adds or refreshes rows in the DataSet to match those in the database. |
|  FillSchema (inherited from DbDataAdapter) | Overloaded. Overridden. Adds a DataTable to a DataSet and configures the schema to match that in the database. |
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  GetBatchedParameter (inherited from DbDataAdapter) | Returns a IDataParameter from one of the commands in the current batch. |
|  GetService (inherited from Component) | Returns an object that represents a service provided by the Component or by its Container. |
|  InitializeBatching (inherited from DbDataAdapter) | Initializes batching for the DbDataAdapter. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

| Name | Description |
|---|---|
|  OnFillError (inherited from DbDataAdapter) | Raises the FillError event. |
|  OnRowUpdated | Overridden. Raises the RowUpdated event using a RowUpdatedEventArgs object. |
|  OnRowUpdating | Overridden. Raises the RowUpdating event using a RowUpdatingEventArgs object, whether or not the update operation is successful. |
|  ShouldSerializeTableMappings (inherited from DataAdapter) | Determines whether one or more DataTableMapping objects exist and they should be persisted. |
|  TerminateBatching (inherited from DbDataAdapter) | Ends batching for the DbDataAdapter. |

**Reference**

"IfxDataAdapter Class" on page 5-253

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataAdapter.CreateRowUpdatedEvent Method:**

Initializes a new instance of the RowUpdatedEventArgs class, regardless of whether the update is successful.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overrides Protected Function CreateRowUpdatedEvent( _
   ByVal dataRow As DataRow, _
   ByVal command As IDbCommand, _
   ByVal statementType As StatementType, _
   ByVal tableMapping As DataTableMapping _
) As RowUpdatedEventArgs
[C#]
protected override RowUpdatedEventArgs CreateRowUpdatedEvent(
   DataRow dataRow,
   IDbCommand command,
   StatementType statementType,
   DataTableMapping tableMapping
);
[C++]
protected: RowUpdatedEventArgs* CreateRowUpdatedEvent(
   DataRow* dataRow,
   IDbCommand* command,
   StatementType statementType,
   DataTableMapping* tableMapping
);
[JScript]
protected override function CreateRowUpdatedEvent(
   dataRow : DataRow,
```

```
   command : IDbCommand,
   statementType : StatementType,
   tableMapping : DataTableMapping
) : RowUpdatedEventArgs;
```

**Parameters**

**dataRow**
The DataRow used to update the database.

**command**
The IfxCommand executed during the update operation.

**statementType**
Whether the command is an UPDATE, INSERT, DELETE, or SELECT statement.

**tableMapping**
A DataTableMapping object.

**Return value**

A new instance of the System.Data.Common.RowUpdatedEventArgs class.

**Remarks**

**Notes to inheritors:** When overriding CreateRowUpdatedEvent in a derived class, be sure to call the CreateRowUpdatedEvent method of the base class.

**Reference**

"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataAdapter.CreateRowUpdatingEvent Method:**

Initializes a new instance of the RowUpdatingEventArgs class.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overrides Protected Function CreateRowUpdatingEvent( _
   ByVal dataRow As DataRow, _
   ByVal command As IDbCommand, _
   ByVal statementType As StatementType, _
   ByVal tableMapping As DataTableMapping _
) As RowUpdatingEventArgs
[C#]
protected override RowUpdatingEventArgs CreateRowUpdatingEvent(
   DataRow dataRow,
   IDbCommand command,
   StatementType statementType,
   DataTableMapping tableMapping
);
[C++]
protected: RowUpdatingEventArgs* CreateRowUpdatingEvent(
   DataRow* dataRow,
```

```
   IDbCommand* command,
   StatementType statementType,
   DataTableMapping* tableMapping
);
[JScript]
protected override function CreateRowUpdatingEvent(
   dataRow : DataRow,
   command : IDbCommand,
   statementType : StatementType,
   tableMapping : DataTableMapping
) : RowUpdatingEventArgs;
```

**Parameters**

**dataRow**
The DataRow that updates the database.

**command**
The IfxCommand to execute during the update operation.

**statementType**
Whether the command is an UPDATE, INSERT, DELETE, or SELECT statement.

**tableMapping**
A DataTableMapping object.

**Return value**

A new instance of the System.Data.Common.RowUpdatingEventArgs class.

**Remarks**

**Notes to inheritors:** When overriding CreateRowUpdatingEvent in a derived class, be sure to call the CreateRowUpdatingEvent method of the base class.

**Reference**

"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataAdapter.OnRowUpdated Method:**

Raises the RowUpdated event using a RowUpdatedEventArgs object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overrides Protected Sub OnRowUpdated( _
   ByVal value As RowUpdatedEventArgs _
)
[C#]
protected override void OnRowUpdated(
   RowUpdatedEventArgs value
);
[C++]
protected: void OnRowUpdated(
```

```
   RowUpdatedEventArgs* value
);
[JScript]
protected override function OnRowUpdated(
   value : RowUpdatedEventArgs
);
```

**Parameters**

**value** A System.Data.Common.RowUpdatedEventArgs object that contains the event data.

**Remarks**

Raising an event invokes the event handler through a delegate.

**Notes to inheritors:** When overriding OnRowUpdated in a derived class, be sure to call the OnRowUpdated method of the base class.

**Reference**

"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

"IfxDataAdapter.OnRowUpdating Method"

**IfxDataAdapter.OnRowUpdating Method:**

Raises the RowUpdating event using a RowUpdatingEventArgs object, whether or not the update operation is successful.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overrides Protected Sub OnRowUpdating( _
   ByVal value As RowUpdatingEventArgs _
)
[C#]
protected override void OnRowUpdating(
   RowUpdatingEventArgs value
);
[C++]
protected: void OnRowUpdating(
   RowUpdatingEventArgs* value
);
[JScript]
protected override function OnRowUpdating(
   value : RowUpdatingEventArgs
);
```

**Parameters**

**value** A System.Data.Common.RowUpdatingEventArgs object that contains the event data.

**Remarks**

Raising an event invokes the event handler through a delegate.

**Notes to inheritors:** When overriding OnRowUpdating in a derived class, be sure to call the OnRowUpdating method of the base class.

**Reference**

"IfxDataAdapter Class" on page 5-253
"IfxDataAdapter Members" on page 5-254
"IBM.Data.Informix Namespace" on page 5-1
"IfxDataAdapter.OnRowUpdated Method" on page 5-272

## IfxDataAdapter Properties

The properties of the IfxDataAdapter class are listed here. For a complete list of IfxDataAdapter class members, see the IfxDataAdapter Members topic.

### Public Properties

| Name | Description |
|---|---|
| AcceptChangesDuringFill (inherited from DataAdapter) | Gets or sets a value indicating whether AcceptChanges is called on a DataRow after it is added to the DataTable. |
| Container (inherited from Component) | Gets the IContainer that contains the Component. |
| ContinueUpdateOnError (inherited from DataAdapter) | Gets or sets a value that specifies whether to generate an exception, or the row in error when an error is encountered during a row update. |
| DeleteCommand | Gets or sets an SQL statement or stored procedure used to delete records in the database. |
| InsertCommand | Gets or sets an SQL statement or stored procedure used to insert new records into the database. |
| MissingMappingAction (inherited from DataAdapter) | Determines the action to take when incoming data does not have a matching table or column. |
| MissingSchemaAction (inherited from DataAdapter) | Determines the action to take when existing DataSet schema does not match incoming data. |
| SelectCommand | Gets or sets an SQL statement or stored procedure used to select records in the database. |
| Site (inherited from Component) | Gets or sets the ISite of the Component. |
| TableMappings (inherited from DataAdapter) | Gets a collection that provides the master mapping between a source table and a DataTable. |
| UpdateBatchSize | Overloaded. Gets or sets the number of commands in a batch to be sent to the database server for execution. |

| Name | Description |
|---|---|
|  UpdateCommand | Gets or sets an SQL statement or stored procedure used to update records in the database. |

## Protected Properties

| Name | Description |
|---|---|
|  DesignMode (inherited from Component) | Gets a value that indicates whether the Component is currently in design mode. |
|  Events (inherited from Component) | Gets the list of event handlers that are attached to this Component. |

**Reference**

"IfxDataAdapter Class" on page 5-253

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataAdapter.DeleteCommand Property:**

Gets or sets an SQL statement or stored procedure used to delete records in the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property DeleteCommand As IfxCommand

[C#]
public new IfxCommand
 DeleteCommand {get; set;}
[C++]
public: __property IfxCommand
* get_DeleteCommand();
public: __property void set_DeleteCommand(IfxCommand
*);
[JScript]
public function get DeleteCommand() : IfxCommand
;
public function set DeleteCommand(IfxCommand
);
```

**Property value**

A IfxCommand used during an update operation to delete records in the database that correspond to deleted rows in the DataSet.

**Remarks**

When the DeleteCommand property is assigned to a previously created IfxCommand , the IfxCommand is not cloned. Instead, the DeleteCommand maintains a reference to the previously created IfxCommand.

During an update operation, if DeleteCommand is not set and primary key information is present in the DataSet, you can use the IfxCommandBuilder class to automatically generate the DeleteCommand, and additional commands needed to reconcile the DataSet to the database. To do this, set the SelectCommand property of the the IfxDataAdapter . The generation logic also requires key column information to be present in the DataSet. For more information see "Automatically Generated Commands" in the Microsoft(R) .NET Framework SDK documentation.

**Example**

[Visual Basic, C#] The following example creates a IfxDataAdapter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxDataAdapter()
     Dim mySelectText As String = _
       "SELECT * FROM STAFF ORDER BY ID"
     Dim myConnString As String = _
       "DATABASE=SAMPLE;"
     Dim myDataAdapter As New IfxDataAdapter(mySelectText, myConnString)
     Dim myDataAdapter.DeleteCommand = New IfxCommand(
       "DELETE FROM STAFF WHERE JOB ='Sales' ", myConnString)
 End Sub

[C#]
public void CreateIfxDataAdapter () {
    string mySelectText = "SELECT * FROM STAFF ORDER BY ID";
    string myConnString = "DATABASE=SAMPLE;";
    IfxDataAdapter myDataAdapter = new IfxDataAdapter(mySelectText,myConnString);
    myDataAdapter.DeleteCommand = New IfxCommand(
      "DELETE FROM STAFF WHERE JOB ='Sales' ", myConnString);
 }
```

**Reference**


“IfxDataAdapter Class” on page 5-253

“IfxDataAdapter Members” on page 5-254

“IBM.Data.Informix Namespace” on page 5-1

“IfxDataAdapter.InsertCommand Property”

“IfxDataAdapter.SelectCommand Property” on page 5-278

“IfxDataAdapter.UpdateCommand Property” on page 5-280


**IfxDataAdapter.InsertCommand Property:**

Gets or sets an SQL statement or stored procedure used to insert new records into the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property InsertCommand As IfxCommand

[C#]
public new IfxCommand
 InsertCommand {get; set;}
[C++]
public: __property IfxCommand
```

```
* get_InsertCommand();
public: __property void set_InsertCommand(IfxCommand
*);
[JScript]
public function get InsertCommand() : IfxCommand
;
public function set InsertCommand(IfxCommand
);
```

**Property value**

A IfxCommand used during an update operation to insert records in the database that correspond to new rows in the DataSet.

**Remarks**

When the InsertCommand property is assigned to a previously created IfxCommand object, the IfxCommand is not cloned. Instead, InsertCommand maintains a reference to the previously created IfxCommand.

During an update operation, if InsertCommand is not set and primary key information is present in the DataSet, you can use the IfxCommandBuilder class to automatically generate InsertCommand, and additional commands needed to reconcile the DataSet to the database. To do this, set the SelectCommand property of the IfxDataAdapter. The generation logic also requires key column information to be present in the DataSet. For more information see "Automatically Generated Commands" in the Microsoft(R) .NET Framework SDK documentation.

**Note:** If execution of this command returns rows, these rows may be added to the DataSet depending upon how you set the IfxCommand.UpdatedRowSource property of the IfxCommand object.

**Example**

[Visual Basic, C#] The following example creates a IfxDataAdapter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxDataAdapter()
     Dim mySelectText As String = _
        "SELECT * FROM STAFF ORDER BY ID"
     Dim myConnString As String = _
        "DATABASE=SAMPLE;"
     Dim myDataAdapter As New IfxDataAdapter(mySelectText, myConnString)
     Dim myDataAdapter.InsertCommand As New IfxCommand(
       "INSERT INTO STAFF
       VALUES ( 360, 'Johnson',84,'Clerk',2,11500.00, 100.00)", myConnString)
End Sub

[C#]
public void CreateIfxDataAdapter ()
{
    string mySelectText = "SELECT * FROM STAFF ORDER BY ID";
    string myConnString = "DATABASE=SAMPLE;";
    IfxDataAdapter myDataAdapter = new IfxDataAdapter(mySelectText,myConnString);
    myDataAdapter.InsertCommand = new IfxCommand(
      "INSERT INTO STAFF
      VALUES ( 360, 'Johnson',84,'Clerk',2,11500.00, 100.00)", myConnString);
}
```

**Reference**

"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254
"IBM.Data.Informix Namespace" on page 5-1
"IfxDataAdapter.DeleteCommand Property" on page 5-275
"IfxDataAdapter.SelectCommand Property"
"IfxDataAdapter.UpdateCommand Property" on page 5-280

**IfxDataAdapter.SelectCommand Property:**

Gets or sets an SQL statement or stored procedure used to select records in the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property SelectCommand As IfxCommand

[C#]
public new IfxCommand
 SelectCommand {get; set;}
[C++]
public: __property IfxCommand
* get_SelectCommand();
public: __property void set_SelectCommand(IfxCommand
*);
[JScript]
public function get SelectCommand() : IfxCommand
;
public function set SelectCommand(IfxCommand
);
```

**Property value**

A IfxCommand that is used during a fill operation to select records from database for placement in the DataSet.

**Remarks**

When SelectCommand is assigned to a previously created IfxCommand, the IfxCommand is not cloned. Instead, the SelectCommand maintains a reference to the previously created IfxCommand object.

If SelectCommand does not return any rows, no tables are added to the DataSet, and no exception is raised.

**Example**

[Visual Basic, C#] The following example creates a IfxDataAdapter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxDataAdapter()
     Dim myConnection As IfxConnection = New IfxConnection("DATABASE=SAMPLE")
     Dim myDataAdapter As New IfxDataAdapter()
     Dim mySelectCommand As New IfxCommand()
     myDataAdapter.SelectCommand.CommandText = _
```

```
        "SELECT * FROM STAFF ORDER BY ID"
 End Sub

[C#]
public void CreateIfxDataAdapter () {
    IfxConnection myConnection = new IfxConnection("DATABASE=SAMPLE");
    IfxDataAdapter myDataAdapter = new IfxDataAdapter();
    IfxCommand mySelectCommand = new IfxCommand();
    myDataAdapter.SelectCommand.CommandText = "SELECT * FROM STAFF ORDER BY ID";
}
```

**Reference**


"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

"IfxDataAdapter.DeleteCommand Property" on page 5-275

"IfxDataAdapter.InsertCommand Property" on page 5-276

"IfxDataAdapter.UpdateCommand Property" on page 5-280


### IfxDataAdapter.UpdateBatchSize Property:

Gets or sets the number of commands in a batch to be sent to the database server for execution.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Property UpdateBatchSize As Integer
[C#]
public override int UpdateBatchSize { get; set; }
[C++]
public:
virtual property int UpdateBatchSize {
   int get () override;
   void set (int value) override;
}
[JScript]
public override function get UpdateBatchSize () : int
public override function set UpdateBatchSize (value : int)
```

**Property value**

Gets or sets an integer value that specifies the number of commands that will be processed in each batch of commands sent to the database server.

| Value | Behavior |
|---|---|
| 0 | The entire update is executed in a single batch. |
| 1 | Disable batch processing. Each update is sent to the database individually. |
| > 1 | The number of commands to execute as a batch before starting a new batch. If this value is greater than the number of rows, the update will be done as a single batch. |

### Example

[C#] The following lines of code result in a batch size of 99.

```
[C#]
    IfxDataAdapter adapter = new IfxDataAdapter();
    adapter.UpdateBatchSize = 99;
```

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

### IfxDataAdapter.UpdateCommand Property:

Gets or sets an SQL statement or stored procedure used to update records in the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Property UpdateCommand As IfxCommand

[C#]
public new IfxCommand
 UpdateCommand {get; set;}
[C++]
public: __property IfxCommand
* get_UpdateCommand();
public: __property void set_UpdateCommand(IfxCommand
*);
[JScript]
public function get UpdateCommand() : IfxCommand
;
public function set UpdateCommand(IfxCommand
);
```

### Property value

A IfxCommand used during an update operation to update records in the database that correspond to modified rows in the DataSet.

### Remarks

When UpdateCommand is assigned to a previously created IfxCommand , the IfxCommand is not cloned. Instead, the UpdateCommand maintains a reference to the previously created IfxCommand object.

During an update operation, if UpdateCommand is not set and primary key information is present in the DataSet, you can use the IfxCommandBuilder class to automatically generate UpdateCommand, and additional commands needed to

reconcile the DataSet to the database. To do this, set the SelectCommand property of the IfxDataAdapter . The generation logic also requires key column information to be present in the DataSet. For more information see "Automatically Generated Commands" in the Microsoft(R) .NET Framework SDK documentation.

**Note:** If execution of this command returns rows, these rows may be merged with the DataSet depending upon how you set the IfxCommand.UpdatedRowSource property of the IfxCommand object.

**Example**

[Visual Basic, C#] The following example creates a IfxDataAdapter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxDataAdapter()
     Dim mySelectText As String = _
        "SELECT * FROM STAFF ORDER BY ID"
     Dim myConn As New IfxConnection _
        ("DATABASE=SAMPLE;")
     Dim myDataAdapter As New IfxDataAdapter(mySelectText, myConn)
     Dim myDataAdapter.UpdateCommand As New IfxCommand(
       "UPDATE STAFF SET DEPT=100 where JOB='Mgr'", myConnString)
 End Sub

[C#]
public void CreateIfxDataAdapter () {
    string mySelectText = "SELECT * FROM STAFF ORDER BY ID";
    IfxConnection myConn = new IfxConnection("DATABASE=SAMPLE;");
    IfxDataAdapter myDataAdapter = new IfxDataAdapter(mySelectText,myConn);
    myDataAdapter.UpdateCommand = new IfxCommand(
      "UPDATE STAFF SET DEPT=100 where JOB='Mgr'", myConnString);
 }
```

**Reference**


"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

"IfxDataAdapter.DeleteCommand Property" on page 5-275

"IfxDataAdapter.InsertCommand Property" on page 5-276

"IfxDataAdapter.SelectCommand Property" on page 5-278


## IfxDataAdapter Events

The events of the IfxDataAdapter class are listed here. For a complete list of IfxDataAdapter class members, see the IfxDataAdapter Members topic.

### Public Events

| Name | Description |
|---|---|
| Disposed (inherited from Component) | Overloaded. Adds an event handler to listen to the Disposed event on the component. |
| FillError (inherited from DbDataAdapter) | Returned when an error occurs during a fill operation. |
| RowUpdated | Occurs during an update operation after a command is executed against the database. |
| RowUpdating | Occurs during Update before a command is executed against the database. |

**Reference**

"IfxDataAdapter Class" on page 5-253

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataAdapter.RowUpdated Event:**

Occurs during an Update operation after a command is executed against the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Event RowUpdated As IfxRowUpdatedEventHandler
[C#]
public event IfxRowUpdatedEventHandler RowUpdated;
[C++]
public: __event IfxRowUpdatedEventHandler * RowUpdated;
```

[JScript]

In JScript(R), you can handle the events defined by a class, but you cannot declare new events.

**Event data**

The event handler receives an argument of type IfxRowUpdatedEventArgs containing data related to this event. The following IfxRowUpdatedEventArgs properties provide information specific to this event.

| Property | Description |
|---|---|
| IfxRowUpdatingEventArgs.Command | Gets the IfxCommand executed when Update is called. |
| Errors (inherited from RowUpdatedEventArgs) | Gets any errors generated by the IBM Data Server Provider for .NET when the Command was executed. |
| RecordsAffected (inherited from RowUpdatedEventArgs) | Gets the number of rows changed, inserted, or deleted by execution of the SQL statement. |
| Row (inherited from RowUpdatedEventArgs) | Gets the DataRow sent through an Update. |
| StatementType (inherited from RowUpdatedEventArgs) | Gets the type of SQL statement executed. |
| Status (inherited from RowUpdatedEventArgs) | Gets the UpdateStatus of the Command. |
| TableMapping (inherited from RowUpdatedEventArgs) | Gets the DataTableMapping sent through an Update. |

### Remarks

When using the Update method, there are two events that occur per data row updated. The order of execution is as follows:

1. The values in the DataRow are moved to the parameter values.
2. The OnRowUpdating event is raised.
3. The command executes.
4. If the UpdateRowSource enumeration is set to FirstReturnedRecord, the first returned result is placed in the DataRow.
5. If there are output parameters, they are placed in the DataRow.
6. The OnRowUpdated event is raised.
7. AcceptChanges is called.

**Reference**

"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataAdapter.RowUpdating Event:

Occurs during an Update operation before a command is executed against the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Event RowUpdating As IfxRowUpdatingEventHandler
[C#]
public event IfxRowUpdatingEventHandler RowUpdating;
[C++]
public: __event IfxRowUpdatingEventHandler* RowUpdating;
```

[JScript]

In JScript(R), you can handle the events defined by a class, but you cannot declare new events.

### Event data

The event handler receives an argument of type IfxRowUpdatingEventArgs containing data related to this event. The following IfxRowUpdatingEventArgs properties provide information specific to this event.

| Property | Description |
|---|---|
| IfxRowUpdatingEventArgs.Command | Gets or sets the IfxCommand to to execute when Update is called. |
| Errors (inherited from RowUpdatingEventArgs) | Gets any errors generated by the IBM Data Server Provider for .NET when the Command executes. |

| Property | Description |
|---|---|
| Row (inherited from RowUpdatingEventArgs) | Gets the DataRow to send through an Update. |
| StatementType (inherited from RowUpdatingEventArgs) | Gets the type of SQL statement to execute. |
| Status (inherited from RowUpdatingEventArgs) | Gets the UpdateStatus of the Command. |
| TableMapping (inherited from RowUpdatingEventArgs) | Gets the DataTableMapping to send through the Update. |

**Remarks**

When using the Update method, there are two events that occur per data row updated. The order of execution is as follows:

1. The values in the DataRow are moved to the parameter values.
2. The OnRowUpdating event is raised.
3. The command executes.
4. If the UpdateRowSource enumeration is set to FirstReturnedRecord, the first returned result is placed in the DataRow.
5. If there are output parameters, they are placed in the DataRow.
6. The OnRowUpdated event is raised.
7. AcceptChanges is called.

**Reference**

"IfxDataAdapter Class" on page 5-253

"IfxDataAdapter Members" on page 5-254

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader Class

Provides a way of reading a forward-only stream of data rows from a database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### .NET Framework 2.0 3.0, 3.5, and 4.0 Inheritance hierarchy

System.Object
  System.MarshalByRefObject
    System.Data.Common.DbDataReader
      IBM.Data.Informix.IfxDataReader

### .NET Framework 2.0 3.0, 3.5, and 4.0 Syntax

```
[Visual Basic]
NotInheritable Public Class IfxDataReader
   Inherits MarshalByRefObject
   Implements IDataReader, IDisposable, IDataRecord, IEnumerable
[C#]
public sealed class IfxDataReader : MarshalByRefObject,
   IDataReader, IDisposable, IDataRecord, IEnumerable
[C++]
public __gc __sealed class IfxDataReader : public
```

```
   MarshalByRefObject, IDataReader, IDisposable, IDataRecord,
   IEnumerable
[JScript]
public class IfxDataReader extends MarshalByRefObject implements
   IDataReader, IDisposable, IDataRecord, IEnumerable
```

## Remarks

To create a IfxDataReader, you must call the IfxCommand.ExecuteReader method of the IfxCommand object, rather than directly using a constructor.

You can concurrently access data from multiple IfxDataReader instances that use the same IfxConnection instance. Each IfxDataReader instance must be associated with its own IfxCommand instance.

Changes made to a result set by another process or thread while data is being read may be visible to the user of the IfxDataReader. However, the precise behavior is timing dependent.

If your application needs to scroll through result sets in multiple directions, or insert, update, and delete rows, you can use a IfxResultSet instance.

IsClosed and RecordsAffected are the only properties that you can call after the IfxDataReader is closed. In some cases, you must call Close before you can call RecordsAffected.

## Example

[Visual Basic, C#] The following example creates a IfxConnection , a IfxCommand , and a IfxDataReader. The example reads through the data, writing it out to the console. Finally, the example closes the IfxDataReader, then the IfxConnection.

```
[Visual Basic]
Public Sub ReadMyData(myConnString As String)
    Dim mySelectQuery As String = "SELECT SALES, SALES_PERSON FROM SALES"
    Dim myConnection As New IfxConnection(myConnString)
    Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
    myConnection.Open()
    Dim myReader As IfxDataReader
    myReader = myCommand.ExecuteReader()
    ' Always call Read before accessing data.
    While myReader.Read()
        Console.WriteLine(myReader.GetInt32(0).ToString() + ", " _
           + myReader.GetString(1))
    End While
    ' always call Close when done reading.
    myReader.Close()
    ' Close the connection when done with it.
    myConnection.Close()
End Sub

[C#]
public void ReadMyData(string myConnString) {
   string mySelectQuery = "SELECT SALES, SALES_PERSON FROM SALES";
   IfxConnection myConnection = new IfxConnection(myConnString);
   IfxCommand myCommand = new IfxCommand(mySelectQuery,myConnection);
   myConnection.Open();
   IfxDataReader myReader;
   myReader = myCommand.ExecuteReader();
   // Always call Read before accessing data.
   while (myReader.Read()) {
      Console.WriteLine(myReader.GetInt32(0) + ", " + myReader.GetString(1));
   }
```

```
    // always call Close when done reading.
    myReader.Close();
    // Close the connection when done with it.
    myConnection.Close();
}
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader Members

IfxDataReader overview

### Public Properties

| Property | Description |
|---|---|
| Depth | Gets a value indicating the depth of nesting for the current row. |
| FieldCount | Gets the number of columns in the current row. |
| HasRows | Gets a value indicating whether the IfxDataReader contains one or more rows. |
| IsClosed | Indicates whether the IfxDataReader is closed. |
| This | Overloaded. Gets the value of a column in its native format.<br><br>In C#, this property is the indexer for the IfxDataReader class. |
| RecordsAffected | Gets the number of rows changed, inserted, or deleted by execution of the SQL statement. |

### Public Methods

| Method | Description |
|---|---|
| Close | Closes the IfxDataReader object. |
| CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |

| Method | Description |
| --- | --- |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetBoolean | Gets the value of the specified column as a Boolean. |
| GetByte | Gets the value of the specified column as a byte. |
| GetBytes | Reads a stream of bytes from the specified column offset into the buffer as an array, starting at the given buffer offset. |
| GetChar | Gets the value of the specified column as a character. |
| GetChars | Reads a stream of characters from the specified column offset into the buffer as an array, starting at the given buffer offset. |
| GetDataTypeName | Gets the name of the source data type. |
| GetDate | Gets the value of the specified column as a Date object. |
| GetDateTime | Gets the value of the specified column as a DateTime object. |
| GetIfxBinary | Creates an instance of a IfxBinary object from the column data. |
| GetIfxBlob | Creates an instance of a IfxBlob object from the column data. |
| GetIfxClob | Creates an instance of a IfxClob object from the column data. |
| GetIfxDate | Creates an instance of a IfxDate object from the column data. |
| GetIfxDecimal | Creates an instance of a IfxDecimal object from the column data. |
| GetIfxDouble | Creates an instance of a IfxDouble object from the column data. |
| GetIfxInt16 | Creates an instance of a IfxInt16 object from the column data. |
| GetIfxInt32 | Creates an instance of a IfxInt32 object from the column data. |
| GetIfxInt64 | Creates an instance of a IfxInt64 object from the column data. |
| GetIfxReal | Creates an instance of a IfxReal object from the column data. |
| GetIfxRowId | Creates an instance of a IfxRowId object from the column data. |
| GetIfxString | Creates an instance of a IfxString object from the column data. |
| GetIfxTime | Creates an instance of a IfxTime object from the column data. |

| Method | Description |
|---|---|
|  GetIfxTimeStamp | Creates an instance of a IfxTimeStamp object from the column data. |
|  GetIfxValue | Creates an Object instance from column data. |
|  GetIfxValues | Gets all the column values for the current row. |
|  GetDecimal | Gets the value of the specified column as a Decimal object. |
|  GetDouble | Gets the value of the specified column as a double-precision floating point number. |
|  GetEnumerator | Returns an enumerator that iterates through the IfxDataReader. |
|  GetFieldType | Gets the Type that is the data type of the object. |
|  GetFloat | Gets the value of the specified column as a single-precision floating-point number. |
|  GetGuid | Not supported. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
|  GetInt16 | Gets the value of the specified column as a 16-bit signed integer. |
|  GetInt32 | Gets the value of the specified column as a 32-bit signed integer. |
|  GetInt64 | Gets the value of the specified column as a 64-bit signed integer. |
|  GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
|  GetName | Gets the name of the specified column. |
|  GetOrdinal | Gets the column ordinal, given the name of the column. |
|  GetSchemaTable | Returns a DataTable that describes the column metadata of the IfxDataReader. |
|  GetStream | Gets the value of the specified column as a System.IO.Stream object. |
|  GetString | Gets the value of the specified column as a string. |
|  GetTime | Gets the value of the specified column as a TimeSpan object. |
|  GetTimeSpan | Gets the value of the specified column as a TimeSpan object. |

| Method | Description |
|---|---|
| GetType (inherited from Object) | Gets the Type of the current instance. |
| GetValue | Gets the value of the column at the specified ordinal in its native format. |
| GetValues | Gets all the attribute columns in the current row. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| IsDBNull | Gets a value indicating whether the column contains non-existent or missing values. |
| NextResult | Advances the the IfxDataReader to the next result, when reading the results of batch SQL statements, or a multiple result-set stored procedure. |
| Read | Advances the IfxDataReader to the next record. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

### Protected Methods

| Method | Description |
|---|---|
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxDataReader Class" on page 5-284

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader Methods

The methods of the IfxDataReader class are listed here. For a complete list of IfxDataReader class members, see the IfxDataReader Members topic.

### Public Methods

| Method | Description |
|---|---|
| Close | Closes the IfxDataReader object. |
| CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetBoolean | Gets the value of the specified column as a Boolean. |

| Method | Description |
|---|---|
| GetByte | Gets the value of the specified column as a byte. |
| GetBytes | Reads a stream of bytes from the specified column offset into the buffer as an array, starting at the given buffer offset. |
| GetChar | Gets the value of the specified column as a character. |
| GetChars | Reads a stream of characters from the specified column offset into the buffer as an array, starting at the given buffer offset. |
| GetDataTypeName | Gets the name of the source data type. |
| GetDate | Gets the value of the specified column as a Date object. |
| GetDateTime | Gets the value of the specified column as a DateTime object. |
| GetIfxBinary | Creates an instance of a IfxBinary object from the column data. |
| GetIfxBlob | Creates an instance of a IfxBlob object from the column data. |
| GetIfxClob | Creates an instance of a IfxClob object from the column data. |
| GetIfxDate | Creates an instance of a IfxDate object from the column data. |
| GetIfxDecimal | Creates an instance of a IfxDecimal object from the column data. |
| GetIfxDouble | Creates an instance of a IfxDouble object from the column data. |
| GetIfxInt16 | Creates an instance of a IfxInt16 object from the column data. |
| GetIfxInt32 | Creates an instance of a IfxInt32 object from the column data. |
| GetIfxInt64 | Creates an instance of a IfxInt64 object from the column data. |
| GetIfxReal | Creates an instance of a IfxReal object from the column data. |
| GetIfxRowId | Creates an instance of a IfxRowId object from the column data. |
| GetIfxString | Creates an instance of a IfxString object from the column data. |
| GetIfxTime | Creates an instance of a IfxTime object from the column data. |
| GetIfxTimeStamp | Creates an instance of a IfxTimeStamp object from the column data. |
| GetIfxValue | Creates an Object instance from column data. |

| Method | Description |
| --- | --- |
|  GetIfxValues | Gets all the column values for the current row. |
|  GetDecimal | Gets the value of the specified column as a Decimal object. |
|  GetDouble | Gets the value of the specified column as a double-precision floating point number. |
|  GetEnumerator | Returns an enumerator that iterates through the IfxDataReader. |
|  GetFieldType | Gets the Type that is the data type of the object. |
|  GetFloat | Gets the value of the specified column as a single-precision floating-point number. |
|  GetGuid | Not supported. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
|  GetInt16 | Gets the value of the specified column as a 16-bit signed integer. |
|  GetInt32 | Gets the value of the specified column as a 32-bit signed integer. |
|  GetInt64 | Gets the value of the specified column as a 64-bit signed integer. |
|  GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
|  GetName | Gets the name of the specified column. |
|  GetOrdinal | Gets the column ordinal, given the name of the column. |
|  GetSchemaTable | Returns a DataTable that describes the column metadata of the IfxDataReader. |
|  GetStream | Gets the value of the specified column as a System.IO.Stream object. |
|  GetString | Gets the value of the specified column as a string. |
|  GetTime | Gets the value of the specified column as a TimeSpan object. |
|  GetTimeSpan | Gets the value of the specified column as a TimeSpan object. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  GetValue | Gets the value of the column at the specified ordinal in its native format. |

| Method | Description |
|---|---|
| GetValues | Gets all the attribute columns in the current row. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| IsDBNull | Gets a value indicating whether the column contains non-existent or missing values. |
| NextResult | Advances the the IfxDataReader to the next result, when reading the results of batch SQL statements, or a multiple result-set stored procedure. |
| Read | Advances the IfxDataReader to the next record. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

## Protected Methods

| Method | Description |
|---|---|
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxDataReader Class" on page 5-284
"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.Close Method:**

Closes the IfxDataReader object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub Close()
[C#]
public void Close();
[C++]
public: void Close();
[JScript]
public function Close();
```

**Remarks**

You must explicitly call the Close method when you are finished using a IfxDataReader or to use the associated IfxCommand for any other purpose.

### Example

[Visual Basic, C#] The following example creates a IfxConnection , a IfxCommand , and a IfxDataReader . The example reads through the data, writing it out to the console. Finally, the example closes the IfxDataReader, then the IfxConnection.

```
[Visual Basic]
Public Sub ReadMyData(myConnString As String)
    Dim mySelectQuery As String = "SELECT ID, NAME FROM STAFF"
    Dim myConnection As New IfxConnection(myConnString)
    Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
    myConnection.Open()
    Dim myReader As IfxDataReader
    myReader = myCommand.ExecuteReader()
    ' Always call Read before accessing data.
    While myReader.Read()
        Console.WriteLine(myReader.GetInt16(0).ToString() + ", " _
           + myReader.GetString(1))
    End While
    ' always call Close when done reading.
    myReader.Close()
    ' Close the connection when done with it.
    myConnection.Close()
End Sub

[C#]
public void ReadMyData(string myConnString)
{
   string mySelectQuery = "SELECT ID, NAME FROM STAFF";
   IfxConnection myConnection = new IfxConnection(myConnString);
   IfxCommand myCommand = new IfxCommand(mySelectQuery,myConnection);
   myConnection.Open();
   IfxDataReader myReader;
   myReader = myCommand.ExecuteReader();
   // Always call Read before accessing data.
   while (myReader.Read()) {
      Console.WriteLine(myReader.GetInt16(0) + ", " + myReader.GetString(1));
   }
   // always call Close when done reading.
   myReader.Close();
   // Close the connection when done with it.
   myConnection.Close();
}
```

#### Reference

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.GetBoolean Method:

Gets the value of the specified column as a Boolean. This method is not supported.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetBoolean( _
   ByVal i As Integer _
) As Boolean
```

```
[C#]
public bool GetBoolean(
   int i
);
[C++]
public: bool GetBoolean(
   int i
);
[JScript]
public function GetBoolean(
   i : int
) : Boolean;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

A Boolean that is the value of the column.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

**Remarks**

Call IsDBNull to check for null values before calling this method.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetByte Method:**

Gets the value of the specified column as a byte. This method is not supported.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetByte( _
   ByVal i As Integer _
) As Byte
[C#]
public byte GetByte(
   int i
);
[C++]
public: unsigned char GetByte(
   int i
);
```

```
[JScript]
public function GetByte(
   i : int
) : Byte;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

The value of the specified column as a byte.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

**Remarks**

Call IsDBNull to check for null values before calling this method.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetBytes Method:**

Reads a stream of bytes from the specified column offset into the buffer as an array, starting at the given buffer offset.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetBytes( _
   ByVal i As Integer, _
   ByVal dataIndex As Long, _
   ByVal buffer() As Byte, _
   ByVal bufferIndex As Integer, _
   ByVal length As Integer _
) As Long
[C#]
public long GetBytes(
   int i,
   long dataIndex,
   byte[] buffer,
   int bufferIndex,
   int length
);
[C++]
public: __int64 GetBytes(
   int i,
   __int64 dataIndex,
```

```
   unsigned char buffer __gc[],
   int bufferIndex,
   int length
);
[JScript]
public function GetBytes(
   i : int,
   dataIndex : long,
   buffer : Byte[],
   bufferIndex : int,
   length : int
) : long;
```

### Parameters

**i** The zero-based column ordinal.

**dataIndex**
The index within the field where the read operation is to begin.

**buffer** The buffer into which to read the stream of bytes.

**bufferIndex**
The index where **buffer** is to begin the write operation.

**length** The number of bytes to read.

### Return value

The actual number of bytes read.

### Exceptions

| Exception type | Condition |
|---|---|
| IfxException | Invalid conversion. |

### Remarks

GetBytes returns the number of available bytes in the field. In most cases this is the exact length of the field. However, the number returned may be less than the true length of the field if GetBytes has already been used to obtain bytes from the field. This may be the case, for example, if the IfxDataReader is reading a BLOB into a buffer. For more information, see the SequentialAccess setting of System.Data.CommandBehavior in the Microsoft(R) .NET Framework SDK documentation.

If you pass a buffer that is a null value, GetBytes returns the length of the field in bytes.

No conversions are performed. The data to be retrieved must be of one of the following types:

- IfxType.Blob

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| Blob | BLOB, BYTE |

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

### IfxDataReader.GetChar Method:

Gets the value of the specified column as a character.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetChar( _
   ByVal i As Integer _
) As Char
[C#]
public char GetChar(
   int i
);
[C++]
public: __wchar_t GetChar(
   int i
);
[JScript]
public function GetChar(
   i : int
) : Char;
```

### Parameters

**i** The zero-based column ordinal.

### Return value

The value of the specified column as a character.

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

### Remarks

When RetrieveXmlInBinaryFormat is set to true, this method will throw an InvalidCastException.

No conversions are performed. The data to be retrieved must be one of the following types:

- IfxType.Char
- IfxType.VarChar
- IfxType.LongVarChar

- IfxType.Clob

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| Char | CHAR |
| VarChar | VARCHAR |
| LongVarChar | LVARCHAR |
| Clob | CLOB, TEXT |

Call IsDBNull to check for null values before calling this method.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetChars Method:**

Reads a stream of characters from the specified column offset into the buffer as an array, starting at the given buffer offset.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetChars( _
   ByVal i As Integer, _
   ByVal dataIndex As Long, _
   ByVal buffer() As Char, _
   ByVal bufferIndex As Integer, _
   ByVal length As Integer _
) As Long
[C#]
public long GetChars(
   int i,
   long dataIndex,
   char[] buffer,
   int bufferIndex,
   int length
);
[C++]
public: __int64 GetChars(
   int i,
   __int64 dataIndex,
   __wchar_t buffer __gc[],
   int bufferIndex,
   int length
);
[JScript]
public function GetChars(
   i : int,
   dataIndex : long,
```

```
   buffer : Char[],
   bufferIndex : int,
   length : int
) : long;
```

**Parameters**

**i** The zero-based column ordinal.

**dataIndex**
The index within the row where the read operation is to begin.

**buffer** The buffer into which to copy data.

**bufferIndex**
The index where **buffer** is to begin the write operation.

**length** The number of characters to read.

**Return value**

The actual number of characters read.

**Exceptions**

| Exception type | Condition |
|---|---|
| IfxException | Invalid conversion. |

**Remarks**

GetChars returns the number of available characters in the field. In most cases this is the exact length of the field. However, the number returned may be less than the true length of the field if GetChars has already been used to obtain characters from the field. This may be the case, for example, if the IfxDataReader is reading a CLOB into a buffer. For more information, see the SequentialAccess setting of System.Data.CommandBehavior in the Microsoft(R) .NET Framework SDK documentation.

If you pass a buffer that is a null value. GetChars returns the length of the field in characters.

No conversions are performed. The data to be retrieved must be one of the following types:

- IfxType.Char
- IfxType.VarChar
- IfxType.LongVarChar
- IfxType.Clob

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| Char | CHAR |
| VarChar | VARCHAR |
| LongVarChar | LVARCHAR |
| Clob | CLOB, TEXT |

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetIfxBinary Method:**

Creates an instance of a IfxBinary object from column data.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetIfxBinary( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxBinary
[C#]
public IBM.Data.IfxTypes.IfxBinary GetIfxBinary (int i)
[C++]
public: IBM.Data.IfxTypes.IfxBinary GetIfxBinary(
   int i
);
[JScript]
public function GetIfxBinary(
   i : int
) : IBM.Data.IfxTypes.IfxBinary;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

A IfxBinary object representing the column value.

**Remarks**

No conversions are performed.

The following table describes the mapping between the return object data type and the data server data type.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284
"IfxBinary Structure" on page 5-4
"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetIfxBlob Method:**

Creates an instance of a IfxBlob object from column data.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetIfxBlob( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxBlob
[C#]
public IBM.Data.IfxTypes.IfxBlob GetIfxBlob (int i)
[C++]
public: IBM.Data.IfxTypes.IfxBlob GetIfxBlob(
   int i
);
[JScript]
public function GetIfxBlob(
   i : int
) : IBM.Data.IfxTypes.IfxBlob;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

A IfxBlob object representing the column value.

**Remarks**

No conversions are performed.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| DB2Blob | BLOB, BYTE |

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetIfxClob Method:**

Creates an instance of a IfxClob object from column data.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetIfxClob( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxClob
[C#]
public IBM.Data.IfxTypes.IfxClob GetIfxClob (int i)
[C++]
public: IBM.Data.IfxTypes.IfxClob GetIfxClob(
   int i
);
[JScript]
public function GetIfxClob(
   i : int
) : IBM.Data.IfxTypes.IfxClob;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

A IfxClob object representing the column value.

**Remarks**

No conversions are performed.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| IfxClob | CLOB, TEXT |

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IfxClob Class" on page 5-75

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetIfxDate Method:**

Creates an instance of a IfxDate object from column data.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetIfxDate( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxDate
[C#]
public IBM.Data.IfxTypes.IfxDate GetIfxDate (int i)
[C++]
public: IBM.Data.IfxTypes.IfxDate GetIfxDate(
   int i
);
[JScript]
public function GetIfxDate(
   i : int
) : IBM.Data.IfxTypes.IfxDate;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

A IfxDate object representing the column value.

**Remarks**

No conversions are performed.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| IfxDate | DATETIME (date precision) |

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |

| Exception type | Condition |
|---|---|
| IfxException | Invalid conversion. |

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDate Structure" on page 5-353

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetIfxDecimal Method:**

Creates an instance of a IfxDecimal object from column data.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetIfxDecimal( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxDecimal
[C#]
public IBM.Data.IfxTypes.IfxDecimal GetIfxDecimal (int i)
[C++]
public: IBM.Data.IfxTypes.IfxDecimal GetIfxDecimal(
   int i
);
[JScript]
public function GetIfxDecimal(
   i : int
) : IBM.Data.IfxTypes.IfxDecimal;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

A IfxDecimal object representing the column value.

**Remarks**

No conversions are performed.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| IfxDecimal | MONEY |

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.GetIfxDouble Method:

Creates an instance of a IfxDouble object from column data.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetIfxDouble( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxDouble
[C#]
public IBM.Data.IfxTypes.IfxDouble GetIfxDouble (int i)
[C++]
public: IBM.Data.IfxTypes.IfxDouble GetIfxDouble(
   int i
);
[JScript]
public function GetIfxDouble(
   i : int
) : IBM.Data.IfxTypes.IfxDouble;
```

### Parameters

**i** The zero-based column ordinal.

### Return value

A IfxDouble object representing the column value.

### Remarks

No conversions are performed.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| IfxDouble | DECIMAL (≤31), DOUBLE PRECISION |

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.GetIfxInt16 Method:

Creates an instance of a IfxInt16 object from column data.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetIfxInt16( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxInt16
[C#]
public IBM.Data.IfxTypes.IfxInt16 GetIfxInt16 (int i)
[C++]
public: IBM.Data.IfxTypes.IfxInt16 GetIfxInt16(
   int i
);
[JScript]
public function GetIfxInt16(
   i : int
) : IBM.Data.IfxTypes.IfxInt16;
```

### Parameters

**i** The zero-based column ordinal.

### Return value

A IfxInt16 object representing the column value.

### Remarks

No conversions are performed.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| IfxInt16 | BOOLEAN, SMALLINT |

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IfxInt16 Structure" on page 5-428

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.GetIfxInt32 Method:

Creates an instance of a IfxInt32 object from column data.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetIfxInt32( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxInt32
[C#]
public IBM.Data.IfxTypes.IfxInt32 GetIfxInt32 (int i)
[C++]
public: IBM.Data.IfxTypes.IfxInt32 GetIfxInt32(
   int i
);
[JScript]
public function GetIfxInt32(
   i : int
) : IBM.Data.IfxTypes.IfxInt32;
```

### Parameters

**i** The zero-based column ordinal.

### Return value

A IfxInt32 object representing the column value.

### Remarks

No conversions are performed.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| IfxInt32 | INT, INTEGER, SERIAL |

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.GetIfxInt64 Method:

Creates an instance of a IfxInt64 object from column data.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetIfxInt64( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxInt64
[C#]
public IBM.Data.IfxTypes.IfxInt64 GetIfxInt64 (int i)
[C++]
public: IBM.Data.IfxTypes.IfxInt64 GetIfxInt64(
   int i
);
[JScript]
public function GetIfxInt64(
   i : int
) : IBM.Data.IfxTypes.IfxInt64;
```

### Parameters

**i** The zero-based column ordinal.

### Return value

A IfxInt64 object representing the column value.

### Remarks

No conversions are performed.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| IfxInt64 | BIGINT, BIGSERIAL, INT8, SERIAL8 |

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.GetIfxReal Method:

Creates an instance of a IfxReal object from column data.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetIfxReal( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxReal
[C#]
public IBM.Data.IfxTypes.IfxReal GetIfxReal (int i)
[C++]
public: IBM.Data.IfxTypes.IfxReal GetIfxReal(
   int i
);
[JScript]
public function GetIfxReal(
   i : int
) : IBM.Data.IfxTypes.IfxReal;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

A IfxReal object representing the column value.

**Remarks**

No conversions are performed.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| IfxReal | REAL, SMALLFLOAT |

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetIfxRowId Method:**

Creates an instance of a IfxRowId object from column data.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetIfxRowId( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxRowId
[C#]
public IBM.Data.IfxTypes.IfxRowId GetIfxRowId (int i)
[C++]
public: IBM.Data.IfxTypes.IfxRowId GetIfxRowId(
   int i
);
```

```
[JScript]
public function GetIfxRowId(
   i : int
) : IBM.Data.IfxTypes.IfxRowId;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

A IfxRowId object representing the column value.

**Remarks**

No conversions are performed.

The following table describes the mapping between the return object data type and the data server data type.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IfxRowId Structure" on page 5-546

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetIfxString Method:**

Creates an instance of a IfxString object from column data.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetIfxString( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxString
[C#]
public IBM.Data.IfxTypes.IfxString GetIfxString (int i)
[C++]
public: IBM.Data.IfxTypes.IfxString GetIfxString(
   int i
);
```

```
[JScript]
public function GetIfxString(
   i : int
) : IBM.Data.IfxTypes.IfxString;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

A IfxString object representing the column value.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IfxString Structure" on page 5-566

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetIfxTime Method:**

Creates an instance of a IfxTime object from column data.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetIfxTime( _
   ByVal i As Integer _
) As IBM.Data.IfxTypes.IfxTime
[C#]
public IBM.Data.IfxTypes.IfxTime GetIfxTime (int i)
[C++]
public: IBM.Data.IfxTypes.IfxTime GetIfxTime(
   int i
);
[JScript]
public function GetIfxTime(
   i : int
) : IBM.Data.IfxTypes.IfxTime;
```

**Parameters**

**i** The zero-based column ordinal.

### Return value

A IfxTime object representing the column value.

### Remarks

No conversions are performed.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| IfxTime | DATETIME (time precision) |

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284
"IfxTime Structure" on page 5-573
"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.GetIfxDateTimeMethod:

Creates an instance of a IfxDateTime object from column data.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetIfxDateTime( _
   ByVal i As Integer _
) As IBM.Data.Informix.IfxDateTime
[C#]
public IBM.Data.Informix.IfxDateTime GetIfxTimeStamp (int i)
[C++]
public: IBM.Data.Informix.IfxDateTime GetIfxTimeStamp(
   int i
);
[JScript]
public function GetIfxDateTime(
   i : int
) : IBM.Data.Informix.IfxDateTime;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

A IfxDateTime object representing the column value.

**Remarks**

No conversions are performed. If the column is of type timestamp with timezone, the time zone value is ignored and a DB2TimeStamp object is returned.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| IfxDateTime | DATETIME (date and time precision) |

The Informix DATETIME data type has the internal format as YYYY-MM-DD HH:MM:SS.nnnnn, whereas the IfxDateTime data type, has a format of YYYY-MM-DD-HH.MM.SS.nnnnnn. This difference in format in the fractional part requires a minor adjustment to match the formats. While reading from the database, a zero will be appended at the least significant digit to match six fractional digits. Similarly while writing to the database, the least significant digit in fraction will be truncated to match the Informix DATETIME type format.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284
"IfxDateTimeStructure" on page 5-584
"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetIfxValue Method:**

Creates an Object instance from column data.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetIfxValue( _
   ByVal i As Integer _
) As Object
[C#]
public object GetIfxValue (int i)
[C++]
public: Object GetIfxValue(
   int i
);
[JScript]
public function GetIfxValue(
   i : int
) : Object;
```

### Parameters

**i** The zero-based column ordinal.

### Return value

An object representing the column value in its native format.

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.GetIfxValues Method:

Gets all the column values for the current row.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetIfxValues( _
   values As Object() _
) As Integer
[C#]
public int GetIfxValues (Object [] values)
[C++]
public: int GetIfxValues(
   Object* values __gc[]
);
[JScript]
public function GetIfxValues(
   values : Object[]
) : int
```

### Parameters

**values** An Object array into which column values for the current row will be copied.

**Return value**

The number of Object instances in the array.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataReader Class" on page 5-284

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetDataTypeName Method:**

Gets the name of the source data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetDataTypeName( _
   ByVal i As Integer _
) As String
[C#]
public string GetDataTypeName(
   int i
);
[C++]
public: String* GetDataTypeName(
   int i
);
[JScript]
public function GetDataTypeName(
   i : int
) : String;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

The name of the source data type.

**Exceptions**

| Exception type | Condition |
|---|---|
| IfxException | Invalid conversion. |

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetDate Method:**

Gets the value of the specified column as a DateTime object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetDate( _
   ByVal i As Integer _
) As DateTime
[C#]
public DateTime GetDate(
   int i
);
[C++]
public: DateTime GetDate(
   int i
);
[JScript]
public function GetDate(
   i : int
) : DateTime;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

A Date object representing the column value.

**Exceptions**

| Exception type | Condition |
|---|---|
| IfxException | Invalid conversion. |

**Remarks**

No conversions are performed. The data to be retrieved must be of IfxType.Date.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| Date | DATETIME (date precision) |

**Reference**

"IfxDataReader Class" on page 5-284
"IfxDataReader Members" on page 5-286
"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetDateTime Method:**

Gets the value of the specified column as a DateTime object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetDateTime( _
   ByVal i As Integer _
) As DateTime
[C#]
public DateTime GetDateTime(
   int i
);
[C++]
public: DateTime GetDateTime(
   int i
);
[JScript]
public function GetDateTime(
   i : int
) : DateTime;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

The value of the specified column as a DateTime object.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

**Remarks**

No conversions are performed. The data to be retrieved must be of IfxType.Date or IfxType.Timestamp.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| Date | DATETIME (date precision) |

Call IsDBNull to check for null values before calling this method.

**Reference**

"IfxDataReader Class" on page 5-284
"IfxDataReader Members" on page 5-286
"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetDecimal Method:**

Gets the value of the specified column as a Decimal object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetDecimal( _
   ByVal i As Integer _
) As Decimal
[C#]
public decimal GetDecimal(
   int i
);
[C++]
public: Decimal GetDecimal(
   int i
);
[JScript]
public function GetDecimal(
   i : int
) : Decimal;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

The value of the specified column as a Decimal object.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

**Remarks**

No conversions are performed. The data to be retrieved must be of DB2Type.Decimal.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| Decimal | MONEY |

Call IsDBNull to check for null values before calling this method.

**Reference**

"IfxDataReader Class" on page 5-284
"IfxDataReader Members" on page 5-286
"IBM.Data.Informix Namespace" on page 5-1

### IfxDataReader.GetDouble Method:

Gets the value of the specified column as a double-precision floating point number.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetDouble( _
   ByVal i As Integer _
) As Double
[C#]
public double GetDouble(
   int i
);
[C++]
public: double GetDouble(
   int i
);
[JScript]
public function GetDouble(
   i : int
) : double;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

The value of the specified column as a double-precision floating point number.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

**Remarks**

No conversions are performed. The data to be retrieved must be of DB2Type.Double.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| Double | DECIMAL (≤31), DOUBLE PRECISION |

Call IsDBNull to check for null values before calling this method.

**Reference**

"IfxDataReader Class" on page 5-284
"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

### IfxDataReader.GetEnumerator Method:

Returns an enumerator that iterates through the IfxDataReader.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function GetEnumerator As System.Collections.IEnumerator
[C#]
public override System.Collections.IEnumerator GetEnumerator ()
[C++]
public:
virtual System.Collections.IEnumerator^ GetEnumerator () override
[JScript]
public override function GetEnumerator () : System.Collections.IEnumerator
```

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxDataReader Class" on page 5-284

"IBM.Data.Informix Namespace" on page 5-1

### IfxDataReader.GetFieldType Method:

Gets the Type that is the data type of the object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetFieldType( _
   ByVal i As Integer _
) As Type
[C#]
public Type GetFieldType(
   int i
);
[C++]
public: Type* GetFieldType(
   int i
);
[JScript]
public function GetFieldType(
   i : int
) : Type;
```

**Parameters**

**i** The zero-based column ordinal.

### Return value

The Type that is the data type of the object.

### Exceptions

| Exception type | Condition |
|---|---|
| IfxException | Invalid conversion. |

**Reference**

"IfxDataReader Class" on page 5-284
"IfxDataReader Members" on page 5-286
"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.GetFloat Method:

Gets the value of the specified column as a single-precision floating-point number.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetFloat( _
   ByVal i As Integer _
) As Single
[C#]
public float GetFloat(
   int i
);
[C++]
public: float GetFloat(
   int i
);
[JScript]
public function GetFloat(
   i : int
) : float;
```

### Parameters

**i** The zero-based column ordinal.

### Return value

The value of the specified column as a single-precision floating-point number.

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

### Remarks

No conversions are performed. The data to be retrieved must be of IfxType.Real.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| Real | REAL, SMALLFLOAT |

Call IsDBNull to check for null values before calling this method.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

### IfxDataReader.GetGuid Method:

Gets the value of the specified column as a globally-unique identifier (GUID). This method is not supported.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetGuid( _
   ByVal i As Integer _
) As Guid
[C#]
public Guid GetGuid(
   int i
);
[C++]
public: Guid GetGuid(
   int i
);
[JScript]
public function GetGuid(
   i : int
) : Guid;
```

### Parameters

**i** The zero-based column ordinal.

### Return value

The value of the specified column as a GUID.

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |

| Exception type | Condition |
|---|---|
| IfxException | Invalid conversion. |

### Remarks

Call IsDBNull to check for null values before calling this method.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.GetInt16 Method:

Gets the value of the specified column as a 16-bit signed integer.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetInt16( _
   ByVal i As Integer _
) As Short
[C#]
public short GetInt16(
   int i
);
[C++]
public: short GetInt16(
   int i
);
[JScript]
public function GetInt16(
   i : int
) : Int16;
```

### Parameters

**i** The zero-based column ordinal.

### Return value

The value of the specified column as a 16-bit signed integer.

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

### Remarks

No conversions are performed. The data to be retrieved must be of DB2Type.SmallInt.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| SmallInt | BOOLEAN, SMALLINT |

Call IsDBNull to check for null values before calling this method.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

### IfxDataReader.GetInt32 Method:

Gets the value of the specified column as a 32-bit signed integer.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetInt32( _
   ByVal i As Integer _
) As Integer
[C#]
public int GetInt32(
   int i
);
[C++]
public: int GetInt32(
   int i
);
[JScript]
public function GetInt32(
   i : int
) : int;
```

### Parameters

**i** The zero-based column ordinal.

### Return value

The value of the specified column as a 32-bit signed integer.

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |

| Exception type | Condition |
|---|---|
| IfxException | Invalid conversion. |

**Remarks**

No conversions are performed. The data to be retrieved must be of DB2Type.Integer.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| Integer | INT, INTEGER, SERIAL |

Call IsDBNull to check for null values before calling this method.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetInt64 Method:**

Gets the value of the specified column as a 64-bit signed integer.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetInt64( _
   ByVal i As Integer _
) As Long
[C#]
public long GetInt64(
   int i
);
[C++]
public: __int64 GetInt64(
   int i
);
[JScript]
public function GetInt64(
   i : int
) : long;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

The value of the specified column as a 64-bit signed integer.

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

### Remarks

No conversions are performed. The data to be retrieved must be of DB2Type.BigInt.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| BigInt | INT8, SERIAL8 |

Call IsDBNull to check for null values before calling this method.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

### IfxDataReader.GetName Method:

Gets the name of the specified column.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetName( _
   ByVal i As Integer _
) As String
[C#]
public string GetName(
   int i
);
[C++]
public: String* GetName(
   int i
);
[JScript]
public function GetName(
   i : int
) : String;
```

### Parameters

**i** The zero-based column ordinal.

**Return value**

A string that is the name of the specified column.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

"IfxDataReader.GetOrdinal Method"

**IfxDataReader.GetOrdinal Method:**

Gets the column ordinal, given the name of the column.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetOrdinal( _
   ByVal value As String _
) As Integer
[C#]
public int GetOrdinal(
   string value
);
[C++]
public: int GetOrdinal(
   String* value
);
[JScript]
public function GetOrdinal(
   value : String
) : int;
```

**Parameters**

**value** The name of the column.

**Return value**

The zero-based column ordinal.

**Remarks**

GetOrdinal performs a case-sensitive lookup first. If it fails, a second case-insensitive search is made.

GetOrdinal is kana-width insensitive.

Because ordinal-based lookups are more efficient than named lookups, it is inefficient to call GetOrdinal within a loop. Instead, call GetOrdinal once and then assign the results to an integer variable for use within the loop.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1
"IfxDataReader.GetName Method" on page 5-327

**IfxDataReader.GetSchemaTable Method:**

Returns a DataTable that describes the column metadata of the IfxDataReader.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetSchemaTable() As DataTable
[C#]
public DataTable GetSchemaTable();
[C++]
public: DataTable* GetSchemaTable();
[JScript]
public function GetSchemaTable() : DataTable;
```

**Return value**

A DataTable that describes the column metadata.

**Exceptions**

| Exception type | Condition |
|---|---|
| InvalidOperationException | The IfxDataReader is closed. |

**Remarks**

The GetSchemaTable method returns metadata about each column in the following order.

| Field name | Description |
|---|---|
| ColumnName | The name of the column if it can be determined; this name might not be unique. The name always reflects the most recent name of the column in the current view or command text. If the column name cannot be determined, a null value is returned. |
| ColumnOrdinal | The ordinal of the column. This value is zero for the bookmark column of the row, if any. Other columns are numbered starting with 1. This field cannot contain a null value. |
| ColumnSize | The maximum possible length of a value in the column. For columns that use a fixed-length data type, this value is the size of the data type. |
| NumericPrecision | If the DbType common language runtime data is a numeric data type, the maximum precision of the column; otherwise, null. |

| Field name | Description |
|---|---|
| NumericScale | If the DbType common language runtime data is decimal, the number of digits to the right of the decimal point; otherwise, null. |
| DataType | A data type that maps to the DbType common language runtime data type. |
| ProviderType | The IfxType enumeration. |
| IsLong | `true` if the column contains a binary large object (BLOB) that contains very long data. |
| AllowDBNull | `true` if the consumer can set the column to a null value or if the driver cannot determine whether the consumer can set the column to a null value; otherwise, `false`. A column can contain a null value, even if it cannot be set to a null value. |
| IsReadOnly | `true` if the column cannot be modified; otherwise, `false`. |
| IsRowVersion | Set if the column contains a persistent row identifier that cannot be written to. |
| IsUnique | `true` or `false`, defined as follows:<br>• `true`. No two rows in the base table, that is returned in BaseTableName, can have the same value in this column. The value of th IsUnique column is guaranteed to be `true` if the column constitutes a key by itself or if there is a constraint of type UNIQUE that applies only to this column.<br>• `false`. The column in the base table can contain duplicate values. The default value for the IsUnique column is `false`. For tables with composite primary keys, the value of the IsUnique column is reported as `false` even if a unique constraint exists on one or more of the columns. |
| IsKey | `true` or `false`, defined as follows:<br>• `true`. The column is one of a set of columns in the rowset that, when taken together, uniquely identify the row in the rowset. This set of columns does not have to be a minimal set of columns. This set of columns can be generated from a base table primary key, a unique constraint, or a unique index.<br>• `false`. The column is not required to uniquely identify the row. |
| IsAutoIncrement | `true` if the column assigns values to new rows in fixed increments; otherwise, `false`. The default for this column is `false`. |

| Field name | Description |
|---|---|
| BaseSchemaName | The name of the schema in the database that contains the column if the base schema name can be determined. The value of BaseSchemaName is a null value if the base schema name cannot be determined. The default for this column is a null value. |
| BaseCatalogName | The name of the catalog in the database that contains the column if the catalog name can be determined. The value of BaseCatalogName is a null value if the base catalog name cannot be determined. The default for this column is a null value. |
| BaseTableName | The name of the table or view in the database that contains the column if the name of the table can be determined. The value of BaseTableName is a null value if the base table name cannot be determined. The default for this column is a null value. |
| BaseColumnName | The name of the column in the database, if possible. This name might be different than the column name returned in the ColumnName column if you used an alias. The value of the BaseColumnName column is a null value if the base column name cannot be determined or if the rowset column is derived from, but is not identical to, a column in the database. The default for the BaseColumnName column is a null value. |

A row is returned for every column in the result set.

IfxCommandBuilder needs to correctly identify the primary keys of a table to work correctly. If the BaseTableName column is not returned for every column in the results of a query, IBM Data Server Provider for .NET tries to parse the SQL statement to find the table names involved in the query. This approach works with UPDATE, INSERT, DELETE and simple SELECT statements but not with stored procedures or SELECT statements that are based on joins. If some or all of the schema information is missing from a table, the IfxCommandBuilder does not work correctly because it does not have enough schema information to automatically generate the correct INSERT, UPDATE, or DELETE statements.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetStream Method:**

Gets the value of the specified XML column as a System.IO.Stream object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetStream( _
   ByVal i As Integer _
) As Stream
[C#]
public Stream GetStream(
   int i
);
[C++]
public: Stream* GetStream(
   int i
);
[JScript]
public function GetStream(
   i : int
) : Stream;
```

### Parameters

**i** The zero-based column ordinal.

### Return value

The value of the specified XML column as a Stream object.

### Exceptions

| Exception type | Condition |
|---|---|
| InvalidCastException | The specified cast is not valid. |
| IfxException | Invalid conversion. |

### Remarks

No conversions are performed. The data to be retrieved must be of the IfxType.Xml type.

The following table describes the mapping between the return object data type and the data server data type.

Call IsDBNull to check for null values before calling this method.

### Version information

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxDataReader Class" on page 5-284
"IfxDataReader Members" on page 5-286
"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.GetString Method:

Gets the value of the specified column as a string.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetString( _
   ByVal i As Integer _
) As String
[C#]
public string GetString(
   int i
);
[C++]
public: String* GetString(
   int i
);
[JScript]
public function GetString(
   i : int
) : String;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

The value of the specified column as a string.

**Reference**

"IfxDataReader Class" on page 5-284
"IfxDataReader Members" on page 5-286
"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetTime Method:**

Gets the value of the specified column as a Time object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetTime( _
   ByVal i As Integer _
) As TimeSpan
[C#]
public TimeSpan GetTime(
   int i
);
[C++]
public: TimeSpan GetTime(
   int i
);
[JScript]
public function GetTime(
   i : int
) : TimeSpan;
```

**Parameters**

**i** The zero-based column ordinal.

**Exceptions**

| Exception type | Condition |
|---|---|
| IfxException | Invalid conversion. |

**Remarks**

No conversions are performed. The data to be retrieved must be of DB2Type.Time.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| Time | DATETIME (time precision) |

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.GetTimeSpan Method:**

Gets the value of the specified column as a TimeSpan object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetTimeSpan( _
   ByVal i As Integer _
) As TimeSpan
[C#]
public TimeSpan GetTimeSpan(
   int i
);
[C++]
public: TimeSpan GetTimeSpan(
   int i
);
[JScript]
public function GetTimeSpan(
   i : int
) : TimeSpan;
```

**Parameters**

**i** The zero-based column ordinal.

### Exceptions

| Exception type | Condition |
|---|---|
| IfxException | Invalid conversion. |

### Remarks

No conversions are performed. The data to be retrieved must be of DB2Type.Time.

The following table describes the mapping between the return object data type and the data server data type.

| IfxType Data Type | Informix Data Type |
|---|---|
| Time | DATETIME (time precision) |

**Reference**

"IfxDataReader Class" on page 5-284

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.GetValue Method:

Gets the value of the column at the specified ordinal in its native format.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Function GetValue( _
   ByVal i As Integer _
) As Object
[C#]
public object GetValue(
   int i
);
[C++]
public: Object* GetValue(
   int i
);
[JScript]
public function GetValue(
   i : int
) : Object;
```

### Parameters

**i** The zero-based column ordinal.

### Return value

The value of the column in its native format.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

### IfxDataReader.GetValues Method:

Gets all the attribute columns in the current row.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function GetValues( _
   ByVal values() As Object _
) As Integer
[C#]
public int GetValues(
   object[] values
);
[C++]
public: int GetValues(
   Object* values __gc[]
);
[JScript]
public function GetValues(
   values : Object[]
) : int;
```

**Parameters**

**values** An array of type Object into which to copy the attribute columns.

**Return value**

The number of instances of Object in the array.

**Reference**

"IfxDataReader Class" on page 5-284
"IfxDataReader Members" on page 5-286
"IBM.Data.Informix Namespace" on page 5-1

### IfxDataReader.IsDBNull Method:

Gets a value indicating whether the column contains non-existent or missing values.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function IsDBNull( _
   ByVal i As Integer _
) As Boolean
[C#]
public bool IsDBNull(
   int i
```

```
);
[C++]
public: bool IsDBNull(
   int i
);
[JScript]
public function IsDBNull(
   i : int
) : Boolean;
```

**Parameters**

**i** The zero-based column ordinal.

**Return value**

`true` if the specified column value is equivalent to DBNull; otherwise, `false`.

**Remarks**

To avoid raising an error, call this method to check for null column values before calling the typed Get methods

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.NextResult Method:**

Advances the the IfxDataReader to the next result, when reading the results of batch SQL statements or a multiple-result-set stored procedure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function NextResult() As Boolean
[C#]
public bool NextResult();
[C++]
public: bool NextResult();
[JScript]
public function NextResult() : Boolean;
```

**Return value**

`true` if there are more result sets; otherwise, `false`.

**Remarks**

Used to process multiple results, which can be generated by executing batch SQL statements or a multiple-result-set stored procedure.

By default, the IfxDataReader is positioned on the first result.

**Reference**

"IfxDataReader Class" on page 5-284
"IfxDataReader Members" on page 5-286
"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.Read Method:**

Advances the IfxDataReader to the next record.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Function Read() As Boolean
[C#]
public bool Read();
[C++]
public: bool Read();
[JScript]
public function Read() : Boolean;
```

**Return value**

`true` if there are more rows; otherwise, `false`.

**Remarks**

The default position of the IfxDataReader is prior to the first record. Therefore, you must call Read before any data.

You can concurrently read data from multiple IfxDataReader instances that use the same IfxConnection instance. Each IfxDataReader instance must be associated with its own IfxCommand instance.

**Example**

[Visual Basic, C#] The following example creates a IfxConnection, a IfxCommand, and a IfxDataReader . The example reads through the data, writing it out to the console. Finally, the example closes the IfxDataReader, then the IfxConnection.

```
[Visual Basic]
Public Sub ReadMyData(myConnString As String)
    Dim mySelectQuery As String = "SELECT ID, NAME FROM STAFF"
    Dim myConnection As New IfxConnection(myConnString)
    Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
    myConnection.Open()
    Dim myReader As IfxDataReader
    myReader = myCommand.ExecuteReader()
    ' Always call Read before accessing data.
    While myReader.Read()
        Console.WriteLine(myReader.GetInt16(0).ToString() + ", " _
           + myReader.GetString(1))
    End While
    ' always call Close when done reading.
    myReader.Close()
    ' Close the connection when done with it.
    myConnection.Close()
End Sub
```

```
[C#]
public void ReadMyData(string myConnString)
{
   string mySelectQuery = "SELECT ID, NAME FROM STAFF";
   IfxConnection myConnection = new IfxConnection(myConnString);
   IfxCommand myCommand = new IfxCommand(mySelectQuery,myConnection);
   myConnection.Open();
   IfxDataReader myReader;
   myReader = myCommand.ExecuteReader();
   // Always call Read before accessing data.
   while (myReader.Read()) {
      Console.WriteLine(myReader.GetInt16(0) + ", " + myReader.GetString(1));
   }
   // always call Close when done reading.
   myReader.Close();
   // Close the connection when done with it.
   myConnection.Close();
}
```

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader Properties

The properties of the IfxDataReader class are listed here. For a complete list of IfxDataReader class members, see the IfxDataReader Members topic.

### Public Properties

| Property | Description |
|---|---|
| Depth | Gets a value indicating the depth of nesting for the current row. |
| FieldCount | Gets the number of columns in the current row. |
| HasRows | Gets a value indicating whether the IfxDataReader contains one or more rows. |
| IsClosed | Indicates whether the IfxDataReader is closed. |
| This | Overloaded. Gets the value of a column in its native format.<br><br>In C#, this property is the indexer for the IfxDataReader class. |
| RecordsAffected | Gets the number of rows changed, inserted, or deleted by execution of the SQL statement. |

**Reference**

"IfxDataReader Class" on page 5-284

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.CacheData Property:**

Indicate if the the data stored in the IfxDataReader instance's current cursor position is to be cached.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property CacheData As Boolean
[C#]
public bool CacheData {get; set;}
[C++]
public: __property bool get_CacheData();
public: __property void set_CacheData(bool);
[JScript]
public function get CacheData() : Boolean;
public function set CacheData(Boolean);
```

**Property value**

`true` if the IfxDataReader object's instance is to be cached; otherwise, `false`. The default is `false`.

**Remarks**

If the CacheData property is set to true, the data in the IfxDataReader instance can be retrieved any number of times by any of the access interfaces (for example, GetDB2String or GetInt16).

This property needs to be set before the first retrieval of data from this object, or the data will not be cached.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.Depth Property:**

Gets a value indicating the depth of nesting for the current row.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Depth As Integer
[C#]
public int Depth {get;}
[C++]
public: __property int get_Depth();
[JScript]
public function get Depth() : int;
```

**Property value**

The depth of nesting for the current row.

### Remarks

The outermost table has a depth of zero.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.FieldCount Property:

Gets the number of columns in the current row.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public ReadOnly Property FieldCount As Integer
[C#]
public int FieldCount {get;}
[C++]
public: __property int get_FieldCount();
[JScript]
public function get FieldCount() : int;
```

### Property value

When not positioned in a valid record set, 0; otherwise the number of columns in the current record. The default is -1.

### Exceptions

| Exception type | Condition |
|---|---|
| NotSupportedException | There is no current connection to a database. |

### Remarks

After executing a query that does not return rows, FieldCount returns 0.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataReader.HasRows Property:

Gets a value indicating whether the IfxDataReader contains one or more rows.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property HasRows As Boolean
[C#]
public bool HasRows {get;}
[C++]
public: __property bool get_HasRows();
[JScript]
public function get HasRows() : Boolean;
```

**Property value**

`true` if the IfxDataReader contains one or more rows; otherwise `false`.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.IsClosed Property:**

Indicates whether the IfxDataReader is closed.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsClosed As Boolean
[C#]
public bool IsClosed {get;}
[C++]
public: __property bool get_IsClosed();
[JScript]
public function get IsClosed() : Boolean;
```

**Property value**

`true` if the IfxDataReader is closed; otherwise, `false`.

**Remarks**

IsClosed and RecordsAffected are the only properties that you can call after the IfxDataReader is closed.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.RecordsAffected Property:**

Gets the number of rows changed, inserted, or deleted by execution of the SQL statement.

**Namespace:**
IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property RecordsAffected As Integer
[C#]
public int RecordsAffected {get;}
[C++]
public: __property int get_RecordsAffected();
[JScript]
public function get RecordsAffected() : int;
```

**Property value**

The number of rows changed, inserted, or deleted. -1 for SELECT statements; 0 if no rows were affected, or the statement failed.

**Remarks**

The RecordsAffected property is not set until all rows are read and you close the IfxDataReader .

The value of this property is cumulative. For example, if two records are inserted in batch mode, the value of RecordsAffected will be 2.

IsClosed and RecordsAffected are the only properties that you can call after the IfxDataReader is closed.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

**IfxDataReader.this Property:**

Gets the value of a column in its native format.

**Overload list**

| Name | Description |
|---|---|
| this(Integer) As Object | Gets the value of the specified column in its native format given the column ordinal. |
| this(String) As Object | Gets the value of the specified column in its native format given the column name. |

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

*IfxDataReader.this (Int32) Property:*

Gets the value of the specified column in its native format given the column ordinal.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Default ReadOnly Property this( _
   ByVal i As Integer _
) As Object
[C#]
public object this[
   int i
] {get;}
[C++]
public: __property Object* get_this(
   int i
);
[JScript]
returnValue = IfxDataReaderObject.this(i);
-or-
returnValue = IfxDataReaderObject(i);
```

**Property value**

The value of the specified column in its native format.

**Exceptions**

| Exception type | Condition |
|---|---|
| IndexOutOfRangeException | The index passed was outside the range of 0 through FieldCount. |

**Reference**

"IfxDataReader Class" on page 5-284
"IfxDataReader Members" on page 5-286
"IBM.Data.Informix Namespace" on page 5-1
"IfxDataReader.this Property" on page 5-343

*IfxDataReader.this (String) Property:*

Gets the value of the specified column in its native format given the column name.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Default ReadOnly Property this( _
   ByVal value As String _
) As Object
[C#]
public object this[
   string value
] {get;}
[C++]
```

```
public: __property Object* get_this(
   String* value
);
[JScript]
returnValue = IfxDataReaderObject.this(value);
-or-
returnValue = IfxDataReaderObject(value);
```

**Property value**

The value of the specified column in its native format.

**Exceptions**

| Exception type | Condition |
|---|---|
| IndexOutOfRangeException | No column with the specified name was found. |

**Remarks**

A case-sensitive lookup is performed first. If it fails, a second case-insensitive search is made.

This method is kana-width insensitive.

**Reference**

"IfxDataReader Class" on page 5-284

"IfxDataReader Members" on page 5-286

"IBM.Data.Informix Namespace" on page 5-1

"IfxDataReader.this Property" on page 5-343

# IfxDataSourceEnumerator Class

Use this class to discover the visible DB2 family data sources.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## .NET Framework 2.0 3.0, 3.5, and 4.0 Inheritance hierarchy

System.Object
  System.Data.Common.DbDataSourceEnumerator
    IBM.Data.Informix.IfxDataSourceEnumerator

## .NET Framework 2.0 3.0, 3.5, and 4.0 Syntax

```
[Visual Basic]
Public NotInheritable Class IfxDataSourceEnumerator
 Inherits DbDataSourceEnumerator
[C#]
public sealed class IfxDataSourceEnumerator : DbDataSourceEnumerator
[C++]
public ref class IfxDataSourceEnumerator sealed : public DbDataSourceEnumerator
[JScript]
public final class IfxDataSourceEnumerator extends DbDataSourceEnumerator
```

## Example

[C#] The following example demonstrates how to create a IfxDataSourceEnumerator instance (with the Instance field), and get a list of all the visible DB2 family data sources (with the GetDataSources method).

```
[C#]
    IfxDataSourceEnumerator dsenum = IfxDataSourceEnumerator.Instance;
    DataTable table = dsenum.GetDataSources();
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataSourceEnumerator Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataSourceEnumerator Members

The members of the IfxDataSourceEnumerator class are listed here.

### Public Methods

| Name | Description |
|---|---|
|  GetDataSources | Overloaded. Returns a DataTable with information about all the visible DB2 family database server instances and databases on those instances. |

### Public Fields

| Name | Description |
|---|---|
|  Instance | Gets a IfxDataSourceEnumerator instance. |

**Reference**

"IfxDataSourceEnumerator Class" on page 5-345

"IBM.Data.Informix Namespace" on page 5-1

## IfxDataSourceEnumerator Fields

The fields of the IfxDataSourceEnumerator class are listed here.

### Public Fields

| Name | Description |
|---|---|
|  Instance | Gets a IfxDataSourceEnumerator instance. |

**Reference**

"IfxDataSourceEnumerator Class" on page 5-345
"IfxDataSourceEnumerator Members" on page 5-346
"IBM.Data.Informix Namespace" on page 5-1

**IfxDataSourceEnumerator.Instance Field:**

Gets a IfxDataSourceEnumerator instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Instance As IfxDataSourceEnumerator
[C#]
public static readonly IfxDataSourceEnumerator Instance
[C++]
public:
static initonly IfxDataSourceEnumerator^ Instance
[JScript]
public static final var Instance () : IfxDataSourceEnumerator
```

**Example**

[C#] The following example demonstrates how to create a IfxDataSourceEnumerator instance (with the Instance field), and get a list of all the visible DB2 family data sources (with the GetDataSources method).

```
[C#]
    IfxDataSourceEnumerator dsenum = IfxDataSourceEnumerator.Instance;
    DataTable table = dsenum.GetDataSources();
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataSourceEnumerator Class" on page 5-345
"IfxDataSourceEnumerator Members" on page 5-346
"IBM.Data.Informix Namespace" on page 5-1

## IfxDataSourceEnumerator Methods

### Public Methods

| Name | Description |
|---|---|
|  GetDataSources | Overloaded. Returns a DataTable with information about all the visible DB2 family database server instances and databases on those instances. |

**Reference**

"IfxDataSourceEnumerator Class" on page 5-345
"IfxDataSourceEnumerator Members" on page 5-346
"IBM.Data.Informix Namespace" on page 5-1

### IfxDataSourceEnumerator.GetDataSources Method:

Returns a DataTable with information about all the visible database server instances and databases on those instances.

#### Overload list

| Name | Description |
|---|---|
| GetDataSources() | Overridden. Returns a DataTable with information about all the visible database server instances and databases on those instances. |
| GetDataSources(Boolean) | Returns a DataTable with information about all the visible database server instances and databases on those instances. A boolean parameter enables the restriction of results to locally cataloged databases. |
| GetDataSources(String Host, String Port, String UID, String PWD) | Enumerates the databases on the instance represented by the Host/Port<br>**Note:** Host being the host system the database instance is running on, and port being the port on which the instance accepts connections.The instance may be either DB2 or Informix. When the instance is DB2, the UID and PWD are ignored. When the instance is Informix the UID and PWD must represent a user capable of connecting to the Informix instance. |

#### Remarks

Using the GetDataSources method, you can find any visible database server instances and databases on those instances. The GetDataSources method works through the IBM Data Server Provider for .NET and is able to see the following information:

- All databases cataloged on the local instance. In the case where there are multiple local instances, only the instance to which the application is attached will be visible.
- All visible remote database server instances and their databases. In order to be visible, the remote database server instances must have a running database administration server (DAS), which is configured to respond to Search and Known Discovery requests.

Database server instances without any databases will not appear in the DataTable. The db2dsdriver configuration file will be used while generating the database, server and alias information.

#### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataSourceEnumerator Class" on page 5-345

"IfxDataSourceEnumerator Members" on page 5-346

"IBM.Data.Informix Namespace" on page 5-1

*IfxDataSourceEnumerator.GetDataSources () Method:*

Returns a DataTable with information about all the visible database server instances and databases on those instances.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function GetDataSources As DataTable
[C#]
public override DataTable GetDataSources ()
[C++]
public:
virtual DataTable^ GetDataSources () override
[JScript]
public override function GetDataSources () : DataTable
```

**Return value**

A DataTable instance, which contains information about the visible database server instances and databases on those instances. The DataTable contains the following columns:

*Table 5-18. Data source enumeration table columns*

| Column name | Description |
|---|---|
| DatabaseAlias | The alias by which the database is known. |
| DatabaseName | The database name. |
| ServerName | Name of the server. This column will contain a null value if the corresponding database exists on a local database server instance. |
| ServiceName | Name of the service through which the database server accepts connections. This column will contain a null value if the corresponding database exists on a local database server instance. |
| InstanceName | The database server instance name. |
| DatabaseLocation | The location of the database (`LOCAL` or `REMOTE`). |

**Remarks**

Using the GetDataSources method, you can find any visible DB2 family database server instances and databases on those instances. The GetDataSources method works through the IBM Data Server Provider for .NET and is able to see the following information:

- all databases cataloged on the local instance. In the case where there are multiple local DB2 instances, GetDataSources behavior is undefined.
- all visible remote database server instances and their databases. In order to be visible, the remote database server instances must have a running database administration server (DAS), which is configured to respond to Search and Known Discovery requests.

Database server instances without any databases will not be visible to the GetDataSources method.

**Example**

[C#] The following example demonstrates how to create a IfxDataSourceEnumerator instance (with the Instance field), and get a list of all the visible DB2 family data sources (with the GetDataSources method).

```
[C#]
    IfxDataSourceEnumerator dsenum = IfxDataSourceEnumerator.Instance;
    DataTable table = dsenum.GetDataSources();
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDataSourceEnumerator Class" on page 5-345

"IfxDataSourceEnumerator Members" on page 5-346

"IBM.Data.Informix Namespace" on page 5-1

*IfxDataSourceEnumerator.GetDataSources (Boolean) Method:*

Returns a DataTable with information about all the visible DB2 family database server instances and databases on those instances. A boolean parameter enables the restriction of results to locally cataloged DB2 family databases.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function GetDataSources ( _
   bLocalDBsOnly As Boolean _
) As DataTable
[C#]
public override DataTable GetDataSources (
   bool bLocalDBsOnly
);
[C++]
public:
virtual DataTable^ GetDataSources (
   bool bLocalDBsOnly
) override
[JScript]
public override function GetDataSources (
   bLocalDBsOnly : Boolean
) : DataTable
```

**Parameters**

**bLocalDBsOnly**
`true` to retrieve information about the local visible DB2 family instance and its databases; `false` to retrieve information about all visible DB2 family database server instances and databases on those instances.

### Return value

A DataTable instance, which contains information about the visible database server instances and databases on those instances. The DataTable contains the following columns:

*Table 5-19. Data source enumeration table columns*

| Column name | Description |
|---|---|
| DatabaseAlias | The alias by which the database is known. |
| DatabaseName | The database name. |
| ServerName | Name of the server. This column will contain a null value if the corresponding database exists on a local database server instance. |
| ServiceName | Name of the service through which the database server accepts connections. This column will contain a null value if the corresponding database exists on a local database server instance. |
| InstanceName | The database server instance name. |
| DatabaseLocation | The location of the database (`LOCAL` or `REMOTE`). |

### Remarks

Using the GetDataSources method, you can find any visible DB2 family database server instances and databases on those instances. The GetDataSources method works through the IBM Data Server Provider for .NET and is able to see the following information:

- all databases cataloged on the local instance. In the case where there are multiple local DB2 instances, GetDataSources behavior is undefined.
- all visible remote database server instances and their databases. In order to be visible, the remote database server instances must have a running database administration server (DAS), which is configured to respond to Search and Known Discovery requests.

Database server instances without any databases will not be visible to the GetDataSources method.

### Example

[C#] The following example demonstrates how to create a IfxDataSourceEnumerator instance (with the Instance field), and get a list of all the local visible DB2 family data sources (with the GetDataSources method).

```
[C#]
    IfxDataSourceEnumerator dsenum =
       IfxDataSourceEnumerator.Instance;
    DataTable table = dsenum.GetDataSources(true);
```

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

### Reference

"IfxDataSourceEnumerator Class" on page 5-345

"IfxDataSourceEnumerator Members" on page 5-346

"IBM.Data.Informix Namespace" on page 5-1

*IfxDataSourceEnumerator.GetDataSources(string, string, string, string) Method:*

Returns a DataTable with information about the specified database server instance and databases on the instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function GetDataSources ( _
   strHost As String _
   strPort As String _
   strUID As String _
   strPWD As String _
) As DataTable
[C#]
public override DataTable GetDataSources (
   string strHost
   string strPort
   string strUID
   string strPWD
);
[C++]
public:
virtual DataTable^ GetDataSources (
   String* strHost,
   String* strPort,
   String* strUID,
   String* strPWD
) override
[JScript]
public override function GetDataSources (
   strHost : String,
   strPort : String,
   strUID : String,
   strPWD : String
) : DataTable
```

**Parameters**

**strHost**
This string can be the name of a system, an IPv4 address, or an IPv6 address. An IPv6 address must be the address only, without the port.

**strPort**
The port through which the database server can be contacted.

**strUID** The user ID to access information from an instance of Informix. No value is required for DB2 data servers.

**strPWD** The password that is associated with the user ID.

**Return value**

A DataTable instance, which contains information about the visible database server instances and databases on those instances. The DataTable contains the following columns:

*Table 5-20. Data source enumeration table columns*

| Column name | Description |
|---|---|
| DatabaseAlias | The alias by which the database is known. |
| DatabaseName | The database name. |
| ServerName | Name of the server. This column will contain a null value if the corresponding database exists on a local database server instance. |
| ServiceName | Name of the service through which the database server accepts connections. This column will contain a null value if the corresponding database exists on a local database server instance. |
| InstanceName | The database server instance name. |
| DatabaseLocation | The location of the database (`LOCAL` or `REMOTE`). |

**Remarks**

Using the GetDataSources method, you can use an IPv4 or IPv6 address to specify a host and identify database server instances and databases on those instances. To be visible, the remote database server instances must have a running database administration server (DAS), which is configured to respond to Search and Known Discovery requests. Database server instances without any databases are visible to the GetDataSources method.

**Example**

[C#] The following example demonstrates how to create a DB2DataSourceEnumerator instance (with the Instance field), and get a list of the visible data sources on 198.17.57.70:1526.

```
[C#]
    IfxDataSourceEnumerator dsenum = IfxDataSourceEnumerator.Instance;
    DataTable table = dsenum.GetDataSources("198.17.57.70", "1526",
        "Admin04", "Admin04PWD");
```

**Version information**

**.NET Framework version**
Supported in: 2.0

# IfxDate Structure

Represents the DATE Informix data type. Encapsulates the DateTime .NET data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
  IBM.Data.Informix.IfxDate

## Syntax

```
[Visual Basic]
Public Structure IfxDate
[C#]
public struct IfxDate
[C++]
public value class IfxDate
```

## Example

[C#] The following example demonstrates how to retrieve a single DATE column value from a table.

```
[C#]
  public static string getParam(IfxConnection conn)
  {
    string mySelectQuery = "SELECT * FROM EMPLOYEE";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);
    IfxDataReader reader = myCommand.ExecuteReader();

    if (reader.Read())
    {
      IfxTime selectValue = reader.GetIfxTime(6);

      if (!selectValue.IsNull) { return selectValue.ToString(); }
    }

    return "NULL";
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDate Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxDate Members

Represents the DATE Informix data type. Encapsulates the DateTime .NET data type. The following tables list the members exposed by the IfxDate class.

### Public Fields

| Field | Description |
|---|---|
| MaxValue | Maximum value for IfxDate: December 31, 9999. |
| MinValue | Minimum value for IfxDate: January 1, 0001. |
| Null | Null value for IfxDate. |

### Public Constructors

| Constructor | Description |
|---|---|
| IfxDate | Initializes a new IfxDate object with the specified DateTime value. |

### Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxDate object is null. |
| Value | Gets the value stored in the IfxDate object in the form of a DateTime object. |

### Public Methods

| Method | Description |
|---|---|
| Parse | Converts the supplied String to IfxDate. |
| ToString | Returns a string that represents the IfxDate structure. |
| op_explicit | Converts the supplied IfxDate structure to a DateTime value. |
| op_implicit | Converts the supplied DateTime value to IfxDate. |

**Reference**

"IfxDate Structure" on page 5-353
"IBM.Data.Informix Namespace" on page 5-1

## IfxDate Fields

The fields of the IfxDate structure are listed here.

### Public Fields

| Field | Description |
|---|---|
| MaxValue | Maximum value for IfxDate: December 31, 9999. |
| MinValue | Minimum value for IfxDate: January 1, 0001. |
| Null | Null value for IfxDate. |

**Reference**

"IfxDate Members" on page 5-354
"IfxDate Structure" on page 5-353
"IBM.Data.Informix Namespace" on page 5-1

**IfxDate.MaxValue Field:**

Maximum value for IfxDate: December 31, 9999.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MaxValue As IfxDate
[C#]
public static readonly IfxDate MaxValue
[C++]
public:
static initonly IfxDate MaxValue
[JScript]
public static final var MaxValue () : IfxDate
```

**Remarks**

The value of this constant is `December 31, 9999`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDate Structure" on page 5-353

"IBM.Data.Informix Namespace" on page 5-1

**IfxDate.MinValue Field:**

Minimum value for IfxDate: January 1, 0001.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MinValue As IfxDate
[C#]
public static readonly IfxDate MinValue
[C++]
public:
static initonly IfxDate MinValue
[JScript]
public static final var MinValue () : IfxDate
```

**Remarks**

The value of this constant is: `January 1, 0001`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDate Structure" on page 5-353

"IBM.Data.Informix Namespace" on page 5-1

**IfxDate.Null Field:**

Null value for IfxDate.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxDate
[C#]
public static readonly IfxDate Null
[C++]
public:
static initonly IfxDate Null
[JScript]
public static final var Null () : IfxDate
```

**Remarks**

The value of this constant is `NULL`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDate Structure" on page 5-353

"IBM.Data.Informix Namespace" on page 5-1

## IfxDate Constructor

Initializes a new IfxDate structure with the specified DateTime value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New(value as DateTime)
[C#]
public IfxDate(DateTime value);
[C++]
public: IfxDate(DateTime value);
[JScript]
public function IfxDate(value : DateTime);
```

### Parameters

**value** A DateTime value to populate the IfxDate instance.

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDate Structure" on page 5-353

"IBM.Data.Informix Namespace" on page 5-1

## IfxDate Methods

The methods of the IfxDate structure are listed here.

### Public Methods

| Method | Description |
|---|---|
|  Parse | Converts the supplied String to IfxDate. |
|  ToString | Returns a string that represents the IfxDate structure. |
|  op_explicit | Converts the supplied IfxDate structure to a DateTime value. |
|  op_implicit | Converts the supplied DateTime value to IfxDate. |

**Reference**

"IfxDate Members" on page 5-354

"IfxDate Structure" on page 5-353

"IBM.Data.Informix Namespace" on page 5-1

**IfxDate.op_explicit Method:**

Converts the supplied IfxDate structure to a DateTime value.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Narrowing Operator CType (source As IfxDate) As DateTime
[C#]
public static explicit operator DateTime (IfxDate source)
[C++]
public:
static explicit operator DateTime (IfxDate source)
```

**Parameters**

**source** A IfxDate structure to be converted to a DateTime value.

**Return value**

A DateTime value converted from a IfxDate instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDate Structure" on page 5-353

"IBM.Data.Informix Namespace" on page 5-1

**IfxDate.op_implicit Method:**

Converts the supplied DateTime value to IfxDate.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Widening Operator CType (source As DateTime) As IfxDate
[C#]
public static implicit operator IfxDate (DateTime source)
[C++]
public:
static implicit operator IfxDate (DateTime source)
```

**Parameters**

**source** A DateTime value to be converted to IfxDate.

**Return value**

A IfxDate structure with the value of **source**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDate Structure" on page 5-353

"IBM.Data.Informix Namespace" on page 5-1

**IfxDate.Parse Method:**

Converts the supplied String to IfxDate.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Function Parse (szDate As String) As IfxDate
[C#]
public static IfxDate Parse (string szDate)
[C++]
public:
```

```
static IfxDate Parse (string szDate)
[JScript]
public static function Parse (szDate String ) : IfxDate
```

**Parameters**

**szDate** A String value to be converted to IfxDate. The string representation of the date must reflect the date format that corresponds to the territory code of the client application.

**Return value**

A IfxDate structure with the numeric value of **szDate**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDate Structure" on page 5-353

"IBM.Data.Informix Namespace" on page 5-1

**IfxDate.ToString Method:**

Returns a string that represents the IfxDate structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxDate structure. The string representation of the date will reflect the date format that corresponds to the territory code of the client application.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDate Structure" on page 5-353

"IBM.Data.Informix Namespace" on page 5-1

## IfxDate Properties

The properties of the IfxDate structure are listed here.

## Public Properties

| Property | Description |
|---|---|
|  IsNull | Gets a value that indicates if the value stored in the IfxDate object is null. |
|  Value | Gets the value stored in the IfxDate object in the form of a DateTime object. |

**Reference**

"IfxDate Members" on page 5-354
"IfxDate Structure" on page 5-353
"IBM.Data.Informix Namespace" on page 5-1

**IfxDate.IsNull Property:**

Gets a value that indicates if the value stored in the IfxDate object is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDate Structure" on page 5-353
"IBM.Data.Informix Namespace" on page 5-1

**IfxDate.Value Property:**

Gets the value stored in the IfxDate structure in the form of a DateTime value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public ReadOnly Property Value As DateTime
[C#]
public DateTime Value {get;}
[C++]
public: __property DateTime get_Value();
[JScript]
public function get Value() : DateTime;
```

### Property value

A DateTime value representing the IfxDate instance.

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDate Structure" on page 5-353

"IBM.Data.Informix Namespace" on page 5-1

## IfxDecimal Structure

Represents the DECIMAL Informix data type. Encapsulates the decimal .NET data type if maxium precision is 29 digits, and the string data type if maximum precision is 31 digits.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Structure IfxDecimal
[C#]
public struct IfxDecimal
[C++]
public value class IfxDecimal
```

### Example

[C#] The following example demonstrates how to retrieve a single DECIMAL column value from a table.

```
[C#]
  public static string getParam(IfxConnection conn)
  {
    string mySelectQuery = "SELECT * FROM EMPLOYEE";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);
    IfxDataReader reader = myCommand.ExecuteReader();

    if (reader.Read())
    {
      IfxDecimal selectValue = reader.GetIfxDecimal(11);

      if (!selectValue.IsNull) { return selectValue.ToString(); }
```

```
    }

    return "NULL";
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxDecimal Members

Represents the DECIMAL Informix data type. Encapsulates the decimal .NET data type if maxium precision is 29 digits, and the string data type if maximum precision is 31 digits. The following tables list the members exposed by the IfxDecimal structure.

### Public Fields

| Field | Description |
|---|---|
| MaxPrecision | Maximum precision for IfxDecimal: 31. |
| MaxScale | Maximum scale for IfxDecimal: 31. |
| MaxValue | The maximum value for IfxDecimal: 9.99999999999999999999999999999999E+30. |
| Null | Null value for IfxDecimal. |
| MinValue | The minimum value for IfxDecimal: -9.99999999999999999999999999999999E+30. |

### Public Constructors

| Constructor | Description |
|---|---|
| IfxDecimal(decimal) | Initializes a new IfxDecimal structure with the specified decimal value. |

### Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxDecimal structure is null. |
| Precision | Gets a value indicating the precision of the IfxDecimal structure's value. |

| Property | Description |
|---|---|
| Scale | Gets a value indicating the scale of the IfxDecimal structure's value. |
| Value | Gets the value stored in the IfxDecimal structure. |
| ValueApproximate | Gets the approximate value stored in the IfxDecimal structure. |

### Public Methods

| Method | Description |
|---|---|
| Parse | Converts the supplied String to IfxDecimal. |
| ToString | Returns a string that represents the IfxDecimal structure. |
| op_explicit | Converts the supplied IfxDecimal structure to a decimal. |
| op_implicit | Converts the supplied decimal to IfxDecimal. |

**Reference**

"IfxDecimal Structure" on page 5-362
"IBM.Data.Informix Namespace" on page 5-1

## IfxDecimal Fields

The fields of the IfxDecimal structure are listed here.

### Public Fields

| Field | Description |
|---|---|
| MaxPrecision | Maximum precision for IfxDecimal: 31. |
| MaxScale | Maximum scale for IfxDecimal: 31. |
| MaxValue | The maximum value for IfxDecimal: 9.99999999999999999999999999999999E+30. |
| Null | Null value for IfxDecimal. |
| MinValue | The minimum value for IfxDecimal: -9.99999999999999999999999999999999E+30. |

**Reference**

"IfxDecimal Members" on page 5-363
"IfxDecimal Structure" on page 5-362
"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.MaxPrecision Field:**

Maximum precision for IfxDecimal: 31.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MaxPrecision As int
[C#]
public static readonly int MaxPrecision
[C++]
public:
static initonly int MaxPrecision
[JScript]
public static final var MaxPrecision () : int
```

**Remarks**

The value of this constant is 31.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362
"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.MaxScale Field:**

Maximum scale for IfxDecimal: 31.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MaxScale As int
[C#]
public static readonly int MaxScale
[C++]
public:
static initonly int MaxScale
[JScript]
public static final var MaxScale () : int
```

**Remarks**

The value of this constant is 31.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.MaxValue Field:**

The maximum value for IfxDecimal: `9.99999999999999999999999999999E+30`.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MaxValue As IfxDecimal
[C#]
public static readonly IfxDecimal MaxValue
[C++]
public:
static initonly IfxDecimal MaxValue
[JScript]
public static final var MaxValue () : IfxDecimal
```

**Remarks**

The value of this constant is `9.99999999999999999999999999999E+30`, with 31 significant figures.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362
"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.MinValue Field:**

The minumum value for IfxDecimal: `-9.99999999999999999999999999999E+30`.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MinValue As IfxDecimal
[C#]
public static readonly IfxDecimal MinValue
[C++]
public:
static initonly IfxDecimal MinValue
[JScript]
public static final var MinValue () : IfxDecimal
```

**Remarks**

The value of this constant is `-9.99999999999999999999999999999E+30`, with 31 significant figures.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.Null Field:**

Null value for IfxDecimal.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxDecimal
[C#]
public static readonly IfxDecimal Null
[C++]
public:
static initonly IfxDecimal Null
[JScript]
public static final var Null () : IfxDecimal
```

**Remarks**

The value of this constant is `NULL`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

## IfxDecimal.IfxDecimal(decimal) Constructor

Initializes a new IfxDecimal structure with the specified decimal value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
Public Sub New(value as Decimal)
[C#]
```

```
public IfxDecimal(decimal value);
[C++]
public: IfxDecimal(Decimal value);
[JScript]
public function IfxDecimal(value : decimal);
```

### Parameters

**value** A decimal value to populate the IfxDecimal instance.

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

## IfxDecimal Methods

The methods of the IfxDecimal structure are listed here.

### Public Methods

| Method | Description |
|---|---|
|  Parse | Converts the supplied String to IfxDecimal. |
|  ToString | Returns a string that represents the IfxDecimal structure. |
|  op_explicit | Converts the supplied IfxDecimal structure to a decimal. |
|  op_implicit | Converts the supplied decimal to IfxDecimal. |

**Reference**

"IfxDecimal Members" on page 5-363

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.op_explicit Method:**

Converts the IfxDecimal structure to a `double`. This method will throw an OverflowException if the precision of the DB2Decimal value is greater than what a .NET decimal type can contain.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Narrowing Operator CType (source As IfxDecimal) As Decimal
[C#]
public static explicit operator decimal (IfxDecimal source)
```

```
[C++]
public:
static explicit operator Decimal (IfxDecimal source)
```

**Parameters**

**source** A IfxDecimal structure to be converted to a decimal.

**Return value**

A decimal value converted from a IfxDecimal instance.

**Exceptions**

| Exception type | Condition |
|---|---|
| IfxNullValueException | The IfxDecimal.Null value cannot be assigned to a decimal. |
| IfxTruncateException | Significant figures are lost when assigning the IfxDecimal value to a decimal. |
| OverflowException | The value being assigned to a decimal is too large. |

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.op_implicit Method:**

Converts the supplied decimal to IfxDecimal.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Widening Operator CType (source As Decimal) As IfxDecimal
[C#]
public static implicit operator IfxDecimal (decimal source)
[C++]
public:
static implicit operator IfxDecimal (Decimal source)
```

**Parameters**

**source** A decimal value to be converted to IfxDecimal.

**Return value**

A IfxDecimal structure with the value of **source**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.Parse Method:**

Converts the supplied String to IfxDecimal.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Function Parse (szDecimal As String) As IfxDecimal
[C#]
public static IfxDecimal Parse (string szDecimal)
[C++]
public:
static IfxDecimal Parse (string szDecimal)
[JScript]
public static function Parse (szDecimal String ) : IfxDecimal
```

**Parameters**

**szDecimal**
A String value to be converted to IfxDecimal.

**Return value**

A IfxDecimal structure with the numeric value of **szDecimal**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.ToString Method:**

Returns a string that represents the IfxDecimal structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxDecimal structure.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

## IfxDecimal Properties

The properties of the IfxDecimal structure are listed here.

### Public Properties

| Property | Description |
|---|---|
|  IsNull | Gets a value that indicates if the value stored in the IfxDecimal structure is null. |
|  Precision | Gets a value indicating the precision of the IfxDecimal structure's value. |
|  Scale | Gets a value indicating the scale of the IfxDecimal structure's value. |
|  Value | Gets the value stored in the IfxDecimal structure. |
|  ValueApproximate | Gets the approximate value stored in the IfxDecimal structure. |

**Reference**

"IfxDecimal Members" on page 5-363

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.IsNull Property:**

Gets a value that indicates if the value stored in the IfxDecimal structure is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

### Property value

`true` if Value is null; otherwise, `false`.

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

## IfxDecimal.Precision Property:

Gets a value indicating the precision of the IfxDecimal structure's value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Property Precision As Integer
[C#]
public int Precision {get;}
[C++]
public: __property int get_Precision();
[JScript]
public function get Precision() : int;
```

### Property value

An integer representing the precision of the IfxDecimal structure. More specifically, this property indicates the number of significant figures of the IfxDecimal value.

### Remarks

An exception is thrown if the IfxDecimal instance has the value IfxDecimal.Null.

### Exceptions

| Exception type | Condition |
|---|---|
| IfxNullValueException | The value of the IfxType is null. |

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.Scale Property:**

Gets a value indicating the scale of the IfxDecimal structure's value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Scale As Integer
[C#]
public int Scale {get;}
[C++]
public: __property int get_Scale();
[JScript]
public function get Scale() : int;
```

**Property value**

An integer representing the scale of the IfxDecimal structure.

**Remarks**

An exception is thrown if the IfxDecimal instance has the value IfxDecimal.Null.

**Exceptions**

| Exception type | Condition |
|---|---|
| IfxNullValueException | The value of the IfxType is null. |

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.ValueApproximate Property:**

Gets the approximate value stored in the IfxDecimal structure. Does not throw an exception if less significant digits are lost.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property ValueApproximate As Decimal
[C#]
public decimal ValueApproximate {get;}
[C++]
public: __property Decimal get_ValueApproximate();
[JScript]
public function get ValueApproximate() : decimal;
```

**Property value**

A decimal representing the approximate value of the IfxDecimal instance.

**Exceptions**

| Exception type | Condition |
|---|---|
| IfxNullValueException | The IfxDecimal.Null value cannot be assigned to a decimal. |
| OverflowException | The value being assigned to a decimal is too large. |

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

**IfxDecimal.Value Property:**

Gets the value stored in the IfxDecimal structure. This method will throw an OverflowException if the precision of the DB2Decimal value is greater than what a .NET decimal type can contain.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Value As Decimal
[C#]
public decimal Value {get;}
[C++]
public: __property Decimal get_Value();
[JScript]
public function get Value() : decimal;
```

**Exceptions**

| Exception type | Condition |
|---|---|
| IfxNullValueException | The IfxDecimal.Null value cannot be assigned to a decimal. |
| IfxTruncateException | Significant figures are lost when assigning the IfxDecimal value to a decimal. |
| OverflowException | The value being assigned to a decimal is too large. |

**Property value**

A decimal representing the IfxDecimal instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDecimal Structure" on page 5-362

"IBM.Data.Informix Namespace" on page 5-1

# IfxDouble Structure

Represents the DOUBLE PRECISION and FLOAT Informix data types. Encapsulates the double .NET data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
Public Structure IfxDouble
[C#]
public struct IfxDouble
[C++]
public value class IfxDouble
```

## Example

[C#] The following example demonstrates how to retrieve a single FLOAT column value from a table.

```
[C#]
  public static string getParam(IfxConnection conn)
  {
    string mySelectQuery = "SELECT DOUBLECOL FROM TESTTBL";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);
    IfxDataReader reader = myCommand.ExecuteReader();

    if (reader.Read())
    {
      IfxDouble selectValue = reader.GetIfxDouble(0);

      if (!selectValue.IsNull) { return selectValue.ToString(); }
```

```
    }

    return "NULL";
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxDouble Members

Represents the DOUBLE PRECISION and FLOAT Informix data types. Encapsulates the double .NET data type. The following tables list the members exposed by the IfxDouble class.

### Public Fields

| Field | Description |
|---|---|
| MaxValue | The maximum value for IfxDouble: 1.79769313486232e308. |
| MinValue | The minumum value for IfxDouble: -1.79769313486232e308. |
| Null | Null value for IfxDouble. |
| Zero | Zero value for IfxDouble. |

### Public Constructors

| Constructor | Description |
|---|---|
| IfxDouble | Initializes a new IfxDouble structure with the specified value. |

### Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxDouble structure is null. |
| Value | Gets the value stored in the IfxDouble structure. |

### Public Methods

| Method | Description |
|---|---|
| Parse | Converts the supplied String to IfxDouble. |
| ToString | Returns a string that represents the IfxDouble structure. |
| op_explicit | Converts the supplied IfxDouble structure to a double. |
| op_implicit | Converts the supplied double to IfxDouble. |

**Reference**

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

## IfxDouble Fields

The fields of the IfxDouble class are listed here.

### Public Fields

| Field | Description |
|---|---|
| MaxValue | The maximum value for IfxDouble: 1.79769313486232e308. |
| MinValue | The minumum value for IfxDouble: -1.79769313486232e308. |
| Null | Null value for IfxDouble. |
| Zero | Zero value for IfxDouble. |

**Reference**

"IfxDouble Members" on page 5-376

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

**IfxDouble.MaxValue Field:**

The maximum value for IfxDouble: 1.79769313486232e308.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MaxValue As IfxDouble
[C#]
public static readonly IfxDouble MaxValue
[C++]
public:
```

```
static initonly IfxDouble MaxValue
[JScript]
public static final var MaxValue () : IfxDouble
```

**Remarks**

The value of this constant is `1.79769313486232e308`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

**IfxDouble.MinValue Field:**

The minumum value for IfxDouble: -1.79769313486232e308.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MinValue As IfxDouble
[C#]
public static readonly IfxDouble MinValue
[C++]
public:
static initonly IfxDouble MinValue
[JScript]
public static final var MinValue () : IfxDouble
```

**Remarks**

The value of this constant is `-1.79769313486232e308`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

**IfxDouble.Null Field:**

Null value for IfxDouble.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxDouble
[C#]
public static readonly IfxDouble Null
[C++]
public:
static initonly IfxDouble Null
[JScript]
public static final var Null () : IfxDouble
```

**Remarks**

The value of this constant is `NULL`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

**IfxDouble.Zero Field:**

Zero value for IfxDouble.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Zero As IfxDouble
[C#]
public static readonly IfxDouble Zero
[C++]
public:
static initonly IfxDouble Zero
[JScript]
public static final var Zero () : IfxDouble
```

**Remarks**

The value of this constant is `0`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

## IfxDouble Constructor

Initializes a new IfxDouble structure with the specified value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New(value as Double)
[C#]
public IfxDouble(double value);
[C++]
public: IfxDouble(double value);
[JScript]
public function IfxDouble(value : double);
```

### Parameters

**value** A double value to populate the IfxDouble instance.

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

## IfxDouble Methods

The methods of the IfxDouble class are listed here.

### Public Methods

| Method | Description |
|---|---|
|  Parse | Converts the supplied String to IfxDouble. |
|  ToString | Returns a string that represents the IfxDouble structure. |
|  op_explicit | Converts the supplied IfxDouble structure to a double. |
|  op_implicit | Converts the supplied double to IfxDouble. |

**Reference**

"IfxDouble Members" on page 5-376

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

**IfxDouble.op_explicit Method:**

Converts the supplied IfxDouble structure to a double.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Narrowing Operator CType (source As IfxDouble) As Double
[C#]
public static implicit operator double (IfxDouble source)
[C++]
public:
static implicit operator double (IfxDouble source)
```

**Parameters**

**source** A IfxDouble structure to be converted to a double.

**Return value**

A double value converted from a IfxDouble instance.

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

**IfxDouble.op_implicit Method:**

Converts the supplied double to IfxDouble.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Widening Operator CType (source As Double) As IfxDouble
[C#]
public static implicit operator IfxDouble (double source)
[C++]
public:
static implicit operator IfxDouble (double source)
```

**Parameters**

**source** A double value to be converted to IfxDouble.

**Return value**

A IfxDouble structure with the value of **source**.

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

### IfxDouble.Parse Method:

Converts the supplied String to IfxDouble.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

#### Syntax

```
[Visual Basic]
Public Shared Function Parse (szDouble As String) As IfxDouble
[C#]
public static IfxDouble Parse (string szDouble)
[C++]
public:
static IfxDouble Parse (string szDouble)
[JScript]
public static function Parse (szDouble String ) : IfxDouble
```

#### Parameters

**szDouble**
A String value to be converted to IfxDouble.

#### Return value

A IfxDouble structure with the numeric value of **szDouble**.

#### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

### IfxDouble.ToString Method:

Returns a string that represents the IfxDouble object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

#### Syntax

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxDouble structure.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

## IfxDouble Properties

The properties of the IfxDouble class are listed here.

### Public Properties

| Property | Description |
|---|---|
|  IsNull | Gets a value that indicates if the value stored in the IfxDouble structure is null. |
|  Value | Gets the value stored in the IfxDouble structure. |

**Reference**

"IfxDouble Members" on page 5-376

"IfxDouble Structure" on page 5-375

"IBM.Data.Informix Namespace" on page 5-1

**IfxDouble.IsNull Property:**

Gets a value that indicates if the value stored in the IfxDouble object is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Structure" on page 5-375
"IBM.Data.Informix Namespace" on page 5-1

**IfxDouble.Value Property:**

Gets the value stored in the IfxDouble object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Value As Double
[C#]
public double Value {get;}
[C++]
public: __property double get_Value();
[JScript]
public function get Value() : double;
```

**Property value**

A `double` representing the IfxDouble instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDouble Structure" on page 5-375
"IBM.Data.Informix Namespace" on page 5-1

# IfxError Class

Collects information relevant to a warning or error returned by the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
  IBM.Data.Informix.IfxError

## Syntax

```
[Visual Basic]
<Serializable>
NotInheritable Public Class IfxError
[C#]
[Serializable]
public sealed class IfxError
[C++]
[Serializable]
public __gc __sealed class IfxError
```

```
[JScript]
public
   Serializable
class IfxError
```

## Remarks

An instance of this class is created whenever an error occurs on a database operation in your application. Each instance of IfxError created by the IfxDataAdapter is then managed by the IfxErrorCollection class, which in turn is created by the IfxException class.

If the severity of the error is too great, the server can close the IfxConnection. If the server does close the connection, the application will need to open a new connection.

## Example

[Visual Basic, C#] The following example displays the properties of the IfxError.

```
[Visual Basic]
Public Sub DisplayIfxErrorCollection(myException As IfxException)
    Dim i As Integer
    For i = 0 To myException.Errors.Count - 1
        MessageBox.Show("Index #" + i.ToString() + ControlChars.Cr _
           + "Message: " + myException.Errors(i).Message + ControlChars.Cr _
           + "Native: " + myException.Errors(i).NativeError.ToString()
                        + ControlChars.Cr _
           + "Source: " + myException.Errors(i).Source + ControlChars.Cr _
           + "SQL: " + myException.Errors(i).SQLState + ControlChars.Cr)
    Next i
End Sub

[C#]
public void DisplayIfxErrorCollection(IfxException myException)
{
   for (int i=0; i < myException.Errors.Count; i++)
   {
      MessageBox.Show("Index #" + i + "\n" +
             "Message: " + myException.Errors[i].Message + "\n" +
             "Native: " + myException.Errors[i].NativeError.ToString() + "\n" +
             "Source: " + myException.Errors[i].Source + "\n" +
             "SQL: " + myException.Errors[i].SQLState + "\n");
   }
}
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxError Members" on page 5-386

"IBM.Data.Informix Namespace" on page 5-1

"IfxErrorCollection Class" on page 5-395

"IfxException Class" on page 5-403

## IfxError Members

IfxError overview

### Public Properties

| Property | Description |
|---|---|
|  Message | Gets a short description of the error. |
|  NativeError | Gets the error information. |
|  RowNumber | Gets an integer value indicating the number of the command in the series of chained commands that caused the error. This property is valid whenever multiple SQL statements are executed as part of a single operation. For example, this property is valid when chaining is active, during a bulk copy operation, and on Update operations by a IfxDataAdapter. |
|  Source | Gets the name of the driver that generated the error. The value is currently IBM.Data.DB2. |
|  SQLState | Gets the five-character error code that follows the ANSI SQL standard for the database. |

### Public Methods

| Method | Description |
|---|---|
|  Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  ToString | Overridden. Gets the complete text of the error message. |

### Protected Methods

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |

| Method | Description |
|---|---|
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxError Class" on page 5-384

"IBM.Data.Informix Namespace" on page 5-1

## IfxError Methods

The methods of the IfxError class are listed here. For a complete list of IfxError class members, see the IfxError Members topic.

### Public Methods

| Method | Description |
|---|---|
|  Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  ToString | Overridden. Gets the complete text of the error message. |

### Protected Methods

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxError Class" on page 5-384

"IBM.Data.Informix Namespace" on page 5-1

**IfxError.ToString Method:**

Gets the complete text of the error message.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Overrides Public Function ToString() As String
[C#]
[Serializable]
public override string ToString();
[C++]
[Serializable]
public: String* ToString();
[JScript]
public
   Serializable
override function ToString() : String;
```

**Return value**

The complete text of the error.

**Example**

[Visual Basic, C#] The following example displays each IfxError within the IfxErrorCollection collection.

```
[Visual Basic]
Public Sub DisplayIfxErrors(myException As IfxException)
    Dim i As Integer
    For i = 0 To myException.Errors.Count - 1
        MessageBox.Show("Index #" + i.ToString() + ControlChars.Cr _
           + "Error: " + myException.Errors(i).ToString() + ControlChars.Cr)
    Next i
End Sub

[C#]
public void DisplayIfxErrors(IfxException myException)
{
   for (int i=0; i < myException.Errors.Count; i++)
   {
      MessageBox.Show("Index #" + i + "\n" +
          "Error: " + myException.Errors[i].ToString() + "\n");
   }
}
```

**Reference**

"IfxError Class" on page 5-384

"IfxError Members" on page 5-386

"IBM.Data.Informix Namespace" on page 5-1

## IfxError Properties

The properties of the IfxError class are listed here. For a complete list of IfxError class members, see the IfxError Members topic.

### Public Properties

| Property | Description |
|---|---|
|  Message | Gets a short description of the error. |

| Property | Description |
|---|---|
|  NativeError | Gets the error information. |
|  RowNumber | Gets an integer value indicating the number of the command in the series of chained commands that caused the error. This property is valid whenever multiple SQL statements are executed as part of a single operation. For example, this property is valid when chaining is active, during a bulk copy operation, and on Update operations by a IfxDataAdapter. |
|  Source | Gets the name of the driver that generated the error. The value is currently IBM.Data.DB2. |
|  SQLState | Gets the five-character error code that follows the ANSI SQL standard for the database. |

**Reference**

"IfxError Class" on page 5-384

"IBM.Data.Informix Namespace" on page 5-1

**IfxError.Message Property:**

Gets a short description of the error.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Public ReadOnly Property Message As String
[C#]
[Serializable]
public string Message {get;}
[C++]
[Serializable]
public: __property String* get_Message();
[JScript]
public
   Serializable
function get Message() : String;
```

**Property value**

A description of the error.

After a server error is detected against an Informix, and an ISAM error code is detected in the SQLCA, the text of that ISAM error is appended to the Message property of the DB2Error object that is generated. A message with ISAM information is formatted as shown in the following example:

```
[IBM][CLI Driver][IDS/UNIX64] Cannot read system catalog (systables)
     (-144 ISAM error: key value locked)
```

For .NET, whenever a server error is detected against and IDS server and an ISAM error code is detected in the SQLCA, the text of that ISAM error will be appended to the Message property of the DB2Error object generated as a result of the error. This is a sample of how a message with ISAM information will be formatted: [IBM][CLI Driver][IDS/UNIX64] Cannot read system catalog (systables) (-144 ISAM error: key value locked) The description of this property will be enhanced to indicate it will include ISAM error information:

**Example**

[Visual Basic, C#] The following example displays the properties of the IfxError .

```
[Visual Basic]
Public Sub DisplayIfxErrorCollection(myException As IfxException)
    Dim i As Integer
    For i = 0 To myException.Errors.Count - 1
        MessageBox.Show("Index #" + i.ToString() + ControlChars.Cr _
           + "Message: " + myException.Errors(i).Message + ControlChars.Cr _
           + "Native: " + myException.Errors(i).NativeError.ToString()
                        + ControlChars.Cr _
           + "Source: " + myException.Errors(i).Source + ControlChars.Cr _
           + "SQL: " + myException.Errors(i).SQLState + ControlChars.Cr)
    Next i
End Sub

[C#]
public void DisplayIfxErrorCollection(IfxException myException)
{
   for (int i=0; i < myException.Errors.Count; i++)
   {
      MessageBox.Show("Index #" + i + "\n" +
             "Message: " + myException.Errors[i].Message + "\n" +
             "Native: " + myException.Errors[i].NativeError.ToString() + "\n" +
             "Source: " + myException.Errors[i].Source + "\n" +
             "SQL: " + myException.Errors[i].SQLState + "\n");
   }
}
```

**Reference**

"IfxError Class" on page 5-384

"IfxError Members" on page 5-386

"IBM.Data.Informix Namespace" on page 5-1

**IfxError.NativeError Property:**

Gets the error information.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Public ReadOnly Property NativeError As Integer
[C#]
[Serializable]
public int NativeError {get;}
[C++]
[Serializable]
public: __property int get_NativeError();
```

```
[JScript]
public
   Serializable
function get NativeError() : int;
```

**Property value**

The error information.

**Example**

[Visual Basic, C#] The following example displays the properties of the IfxError .

```
[Visual Basic]
Public Sub DisplayIfxErrorCollection(myException As IfxException)
    Dim i As Integer
    For i = 0 To myException.Errors.Count - 1
        MessageBox.Show("Index #" + i.ToString() + ControlChars.Cr _
           + "Message: " + myException.Errors(i).Message + ControlChars.Cr _
           + "Native: " + myException.Errors(i).NativeError.ToString()
                        + ControlChars.Cr _
           + "Source: " + myException.Errors(i).Source + ControlChars.Cr _
           + "SQL: " + myException.Errors(i).SQLState + ControlChars.Cr)
    Next i
End Sub

[C#]
public void DisplayIfxErrorCollection(IfxException myException)
{
   for (int i=0; i < myException.Errors.Count; i++)
   {
      MessageBox.Show("Index #" + i + "\n" +
             "Message: " + myException.Errors[i].Message + "\n" +
             "Native: " + myException.Errors[i].NativeError.ToString() + "\n" +
             "Source: " + myException.Errors[i].Source + "\n" +
             "SQL: " + myException.Errors[i].SQLState + "\n");
   }
}
```

**Reference**

"IfxError Class" on page 5-384

"IfxError Members" on page 5-386

"IBM.Data.Informix Namespace" on page 5-1

**IfxError.RowNumber Property:**

Gets an integer value indicating the number of the command in the series of chained commands that caused the error. This property is valid whenever multiple SQL statements are executed as part of a single operation. For example, this property is valid when chaining is active, during a bulk copy operation, and on Update operations by a IfxDataAdapter.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property RowNumber As Integer
[C#]
public int RowNumber {get;}
```

```
[C++]
public: __property int get_RowNumber();
[JScript]
public function get RowNumber() : int;
```

**Property value**

The number of the statement in the series of chained statements that caused the **IfxError**. For example, a **RowNumber** value of 2 means that the second statement in the series of chained statements caused the **IfxError**. This property is valid when chaining is active (when the IfxConnection.Chaining property is true), and also whenever else multiple SQL statements are executed as part of a single operation.

**Example**

[Visual Basic, C#] The following example uses chaining to insert 10000 rows into the STAFF table.

```
[Visual Basic]
Dim con As IfxConnection = new IfxConnection("DATABASE=sample;")
Dim cmd As IfxCommand = con.CreateCommand()
con.Open()

' Initialize an insert statement using parameter markers
cmd.CommandText = "INSERT INTO STAFF(ID) VALUES( ? )"

' Add a parameter
Dim p1 As IfxParameter = cmd.Parameters.Add("@ID", IfxType.Integer )

' Start the chain
con.BeginChain()

Try
   ' Loop to add 10000 rows
   Dim I As Int32
   For I = 1 To 10000
      ' Set the parameter value
      p1.Value = I

      ' Execute the command.
      ' Since chaining is active, this statement is now added
      '   to the chain
      cmd.ExecuteNonQuery()
   Next I

   ' Execute the chain
   con.EndChain()
Catch db2Ex As IfxException
   Dim db2Error As IfxError

   ' Loop through all the errors
   For Each db2Error in db2Ex.Errors
      Console.WriteLine("SQLSTATE =" & db2Error.SQLState )
      Console.WriteLine("NativeErr=" & db2Error.NativeError )
      Console.WriteLine("RowNumber=" & db2Error.RowNumber )
      Console.WriteLine( db2Error.Message )
   Next IfxError
Finally
   ' Explicitly turn chaining off in case it is still on
   If (con.Chaining) Then
      con.EndChain()
   End If
End Try

con.Close()[C#]
```

```
DB2Connection con = new IfxConnection("DATABASE=sample;");
DB2Command cmd = con.CreateCommand();
con.Open();

// Initialize an insert statement using parameter markers
cmd.CommandText = "INSERT INTO STAFF(ID) VALUES( ? )";

// Add a parameter
DB2Parameter p1 = cmd.Parameters.Add("@ID", IfxType.Integer );

// Start the chain
con.BeginChain();

try
{
   // Loop to add 10000 rows
   for( Int32 i = 1; i <= 10000; i++ )
   {
      // Set the parameter value
      p1.Value = i;

      // Execute the command.
      // Since chaining is active, this statement is now added
      //   to the chain
      cmd.ExecuteNonQuery();
   }

   // Execute the chain
   con.EndChain();
}
catch( IfxException db2Ex )
{
   // Loop through all the errors
   foreach( IfxError db2Error in db2Ex.Errors )
   {
      Console.WriteLine("SQLSTATE =" + db2Error.SQLState );
      Console.WriteLine("NativeErr=" + db2Error.NativeError );
      Console.WriteLine("RowNumber=" + db2Error.RowNumber );
      Console.WriteLine( db2Error.Message );
   }
}
finally
{
   // Explicitly turn chaining off in case it is still on
   if( con.Chaining )
   {
      con.EndChain();
   }
}

con.Close();
```

**Reference**

"IfxError Class" on page 5-384

"IfxError Members" on page 5-386

"IBM.Data.Informix Namespace" on page 5-1

**IfxError.SQLState Property:**

Gets the five-character error code that follows the ANSI SQL standard for the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Public ReadOnly Property SQLState As String
[C#]
[Serializable]
public string SQLState {get;}
[C++]
[Serializable]
public: __property String* get_SQLState();
[JScript]
public
   Serializable
function get SQLState() : String;
```

**Property value**

The five-character error code, which identifies the source of the error if the error can be issued from more than one place.

**Example**

[Visual Basic, C#] The following example displays the properties of the IfxError .

```
[Visual Basic]
Public Sub DisplayIfxErrorCollection(myException As IfxException)
    Dim i As Integer
    For i = 0 To myException.Errors.Count - 1
        MessageBox.Show("Index #" + i.ToString() + ControlChars.Cr _
           + "Message: " + myException.Errors(i).Message + ControlChars.Cr _
           + "Native: " + myException.Errors(i).NativeError.ToString()
                        + ControlChars.Cr _
           + "Source: " + myException.Errors(i).Source + ControlChars.Cr _
           + "SQL: " + myException.Errors(i).SQLState + ControlChars.Cr)
    Next i
End Sub

[C#]
public void DisplayIfxErrorCollection(IfxException myException)
{
   for (int i=0; i < myException.Errors.Count; i++)
   {
      MessageBox.Show("Index #" + i + "\n" +
             "Message: " + myException.Errors[i].Message + "\n" +
             "Native: " + myException.Errors[i].NativeError.ToString() + "\n" +
             "Source: " + myException.Errors[i].Source + "\n" +
             "SQL: " + myException.Errors[i].SQLState + "\n");
   }
}
```

**Reference**

"IfxError Class" on page 5-384

"IfxError Members" on page 5-386

"IBM.Data.Informix Namespace" on page 5-1

**IfxError.Source Property:**

Gets the name of the driver that generated the error.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Public ReadOnly Property Source As String
[C#]
[Serializable]
public string Source {get;}
[C++]
[Serializable]
public: __property String* get_Source();
[JScript]
public
   Serializable
function get Source() : String;
```

**Property value**

The name of the driver that generated the error. This value is currently "IBM.Data.Informix".

**Example**

[Visual Basic, C#] The following example displays the properties of a IfxError .

```
[Visual Basic]
Public Sub DisplayIfxErrorCollection(myException As IfxException)
    Dim i As Integer
    For i = 0 To myException.Errors.Count - 1
        MessageBox.Show("Index #" + i.ToString() + ControlChars.Cr _
           + "Message: " + myException.Errors(i).Message + ControlChars.Cr _
           + "Native: " + myException.Errors(i).NativeError.ToString()
                        + ControlChars.Cr _
           + "Source: " + myException.Errors(i).Source + ControlChars.Cr _
           + "SQL: " + myException.Errors(i).SQLState + ControlChars.Cr)
    Next i
End Sub

[C#]
public void DisplayIfxErrorCollection(IfxException myException)
{
   for (int i=0; i < myException.Errors.Count; i++)
   {
      MessageBox.Show("Index #" + i + "\n" +
             "Message: " + myException.Errors[i].Message + "\n" +
             "Native: " + myException.Errors[i].NativeError.ToString() + "\n" +
             "Source: " + myException.Errors[i].Source + "\n" +
             "SQL: " + myException.Errors[i].SQLState + "\n");
   }
}
```

**Reference**

"IfxError Class" on page 5-384

"IfxError Members" on page 5-386

"IBM.Data.Informix Namespace" on page 5-1

## IfxErrorCollection Class

Collects all generated errors.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
IBM.Data.Informix.IfxErrorCollection

## Syntax

```
[Visual Basic]
<Serializable>
NotInheritable Public Class IfxErrorCollection
   Implements ICollection, IEnumerable
[C#]
[Serializable]
public sealed class IfxErrorCollection : ICollection, IEnumerable
[C++]
[Serializable]
public __gc __sealed class IfxErrorCollection : public
   ICollection, IEnumerable
[JScript]
public
   Serializable
class IfxErrorCollection implements ICollection,
   IEnumerable
```

## Remarks

This class is created by IfxException to collect instances of the IfxError class. IfxErrorCollection always contains at least one instance of the IfxError class.

## Example

[Visual Basic, C#] The following example displays each IfxError within the IfxErrorCollection.

```
[Visual Basic]
Public Sub DisplayIfxErrorCollection(myException As IfxException)
    Dim i As Integer
    For i = 0 To myException.Errors.Count - 1
        MessageBox.Show("Index #" + i.ToString() + ControlChars.Cr _
           + "Message: " + myException.Errors(i).Message + ControlChars.Cr _
           + "Native: " + myException.Errors(i).NativeError.ToString()
                        + ControlChars.Cr _
           + "Source: " + myException.Errors(i).Source + ControlChars.Cr _
           + "SQL: " + myException.Errors(i).SQLState + ControlChars.Cr)
    Next i
End Sub

[C#]
public void DisplayIfxErrorCollection(IfxException myException)
{
   for (int i=0; i < myException.Errors.Count; i++)
   {
      MessageBox.Show("Index #" + i + "\n" +
             "Message: " + myException.Errors[i].Message + "\n" +
             "Native: " + myException.Errors[i].NativeError.ToString() + "\n" +
             "Source: " + myException.Errors[i].Source + "\n" +
             "SQL: " + myException.Errors[i].SQLState + "\n");
   }
}
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxErrorCollection Members"

"IBM.Data.Informix Namespace" on page 5-1

"IfxError Class" on page 5-384

"IfxException Class" on page 5-403

## IfxErrorCollection Members

IfxErrorCollection overview

### Public Properties

| Property | Description |
|---|---|
|  Count | Gets the number of errors in the collection. |
|  this | Gets the error at the specified index.<br><br>In C#, this property is the indexer for the IfxErrorCollection class. |

### Public Methods

| Method | Description |
|---|---|
|  CopyTo | Copies the elements of the IfxErrorCollection into an array, starting at the given index within the array. |
|  Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
|  GetEnumerator (inherited from Object) | Overloaded. This member supports the Microsoft(R) .NET Framework infrastructure and is not intended to be used directly from your code. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  ToString (inherited from Object) | Returns a String that represents the current Object. |

### Protected Methods

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxErrorCollection Class" on page 5-395

"IBM.Data.Informix Namespace" on page 5-1

## IfxErrorCollection Methods

The methods of the IfxErrorCollection class are listed here. For a complete list of IfxErrorCollection class members, see the IfxErrorCollection Members topic.

### Public Methods

| Method | Description |
|---|---|
|  CopyTo | Copies the elements of the IfxErrorCollection into an array, starting at the given index within the array. |
|  Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
|  GetEnumerator (inherited from Object) | Overloaded. This member supports the Microsoft(R) .NET Framework infrastructure and is not intended to be used directly from your code. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  ToString (inherited from Object) | Returns a String that represents the current Object. |

### Protected Methods

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |

| Method | Description |
| --- | --- |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxErrorCollection Class" on page 5-395

"IBM.Data.Informix Namespace" on page 5-1

**IfxErrorCollection.CopyTo Method:**

Copies the elements of the IfxErrorCollection into an array, starting at the given index within the array.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
NotOverridable Public Sub CopyTo( _
   ByVal array As Array, _
   ByVal i As Integer _
) Implements ICollection.CopyTo
[C#]
[Serializable]
public void CopyTo(
   Array array,
   int i
);
[C++]
[Serializable]
public: __sealed void CopyTo(
   Array* array,
   int i
);
[JScript]
public
   Serializable
function CopyTo(
   array : Array,
   i : int
);
```

**Implements:**
ICollection.CopyTo

**Parameters**

**array** The array into which to copy the elements.

**i** The starting index of **array**.

**Exceptions**

| Exception type | Condition |
|---|---|
| ArgumentException | The sum of **index** and the number of elements in the IfxErrorCollection is greater than the length of the array. |
| ArgumentNullException | The **array** is a null reference (Nothing in Microsoft(R) Visual Basic(R)). |
| ArgumentOutOfRangeException | The **index** is not valid for the **array**. |

**Reference**

"IfxErrorCollection Class" on page 5-395

"IfxErrorCollection Members" on page 5-397

"IBM.Data.Informix Namespace" on page 5-1

**IfxErrorCollection.GetEnumerator Method:**

This member supports the Microsoft(R) .NET Framework infrastructure and is not intended to be used directly from your code.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
NotOverridable Public Function GetEnumerator() As IEnumerator _
   Implements IEnumerable.GetEnumerator
[C#]
[Serializable]
public IEnumerator GetEnumerator();
[C++]
[Serializable]
public: __sealed IEnumerator* GetEnumerator();
[JScript]
public
   Serializable
function GetEnumerator() : IEnumerator;
```

**Reference**

"IfxErrorCollection Class" on page 5-395

"IfxErrorCollection Members" on page 5-397

"IBM.Data.Informix Namespace" on page 5-1

## IfxErrorCollection Properties

The properties of the IfxErrorCollection class are listed here. For a complete list of IfxErrorCollection class members, see the IfxErrorCollection Members topic.

### Public Properties

| Property | Description |
|---|---|
|  Count | Gets the number of errors in the collection. |

| Property | Description |
|---|---|
|  this | Gets the error at the specified index.<br><br>In C#, this property is the indexer for the IfxErrorCollection class. |

**Reference**

"IfxErrorCollection Class" on page 5-395

"IBM.Data.Informix Namespace" on page 5-1

**IfxErrorCollection.Count Property:**

Gets the number of errors in the collection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Public ReadOnly Property Count As Integer  Implements _
   ICollection.Count
[C#]
[Serializable]
public int Count {get;}
[C++]
[Serializable]
public: __property int get_Count();
[JScript]
public
   Serializable
function get Count() : int;
```

**Implements:**
ICollection.Count

**Property value**

The total number of errors in the collection.

**Example**

[Visual Basic, C#] The following example displays each IfxError within the IfxErrorCollection .

```
[Visual Basic]
Public Sub DisplayIfxErrorCollection(myException As IfxException)
    Dim i As Integer
    For i = 0 To myException.Errors.Count - 1
        MessageBox.Show("Index #" + i.ToString() + ControlChars.Cr _
           + "Message: " + myException.Errors(i).Message
                         + ControlChars.Cr _
           + "Native: " + myException.Errors(i).NativeError.ToString()
              + ControlChars.Cr _
           + "Source: " + myException.Errors(i).Source + ControlChars.Cr _
           + "SQL: " + myException.Errors(i).SQLState + ControlChars.Cr)
    Next i
End Sub
```

```
[C#]
public void DisplayIfxErrorCollection(IfxException myException)
{
   for (int i=0; i < myException.Errors.Count; i++)
   {
      MessageBox.Show("Index #" + i + "\n" +
             "Message: " + myException.Errors[i].Message + "\n" +
             "Native: " + myException.Errors[i].NativeError.ToString() + "\n" +
             "Source: " + myException.Errors[i].Source + "\n" +
             "SQL: " + myException.Errors[i].SQLState + "\n");
   }
}
```

**Reference**

"IfxErrorCollection Class" on page 5-395

"IfxErrorCollection Members" on page 5-397

"IBM.Data.Informix Namespace" on page 5-1

### IfxErrorCollection.Item Property:

Gets the error at the specified index.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Public Default ReadOnly Property Item( _
   ByVal i As Integer _
) As IfxError

[C#]
[Serializable]
public IfxError
 this[
   int i
] {get;}
[C++]
[Serializable]
public: __property IfxError
* get_Item(
   int i
);
[JScript]
returnValue = IfxErrorCollectionObject.Item(i);
-or-
returnValue = IfxErrorCollectionObject(i);
```

**Property value**

A IfxError that contains the error at the specified index.

**Remarks**

The following example displays each IfxError within the IfxErrorCollection .

```
[C#]
public void DisplayIfxErrorCollection(IfxException myException)
 {
   for (int i=0; i < myException.Errors.Count; i++)
```

```
  {
    MessageBox.Show("Index #" + i + "\n" +
      "Message: " + myException.Errors[i].Message + "\n" +
      "Native: " + myException.Errors[i].NativeError.ToString() + "\n" +
      "Source: " + myException.Errors[i].Source + "\n" +
      "SQL: " + myException.Errors[i].SQLState + "\n");
  }
}
```

**Reference**

"IfxErrorCollection Class" on page 5-395

"IfxErrorCollection Members" on page 5-397

"IBM.Data.Informix Namespace" on page 5-1

# IfxException Class

The exception that is generated when a warning or error is returned by a database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## .NET Framework 2.0 3.0, 3.5, and 4.0 Inheritance hierarchy

System.Object
  System.Exception
    System.SystemException
      System.Runtime.InteropServices.SystemException
        System.Data.Common.DbException
          IBM.Data.Informix.IfxException

## .NET Framework 2.0 3.0, 3.5, and 4.0 Syntax

```
[Visual Basic]
<Serializable>
NotInheritable Public Class IfxException
   Inherits DbException
[C#]
[Serializable]
public sealed class IfxException : DbException
[C++]
[Serializable]
public __gc __sealed class IfxException : public DbException
[JScript]
public
   Serializable
class IfxException extends DbException
```

## Remarks

This class is created whenever an error generated by the database server. (Client-side errors are raised as standard common language runtime exceptions.) It always contains at least one instance of IfxError .

If the severity of the error is too great, the server can close the IfxConnection . If the server does close the connection, the application will need to open a new connection.

## Example

[Visual Basic, C#] The following example generates an IfxException due to a missing database, and then displays the exception.

```
[Visual Basic]
Public Sub ThrowIfxException()
    Dim mySelectQuery As String = "SELECT column1 FROM table1"
    Dim myConnection As New IfxConnection _
       ("DATABASE=BADDB;")
    Dim myCommand As New IfxCommand(mySelectQuery, myConnection)
    Try
        myCommand.Connection.Open()
    Catch myException As IfxException
        Dim i As Integer
        For i = 0 To myException.Errors.Count - 1
            MessageBox.Show("Index #" + i.ToString() + ControlChars.Cr _
               + "Message: " + myException.Errors(i).Message
                             + ControlChars.Cr _
               + "Native: " + myException.Errors(i).NativeError.ToString()
                + ControlChars.Cr _
               + "Source: " + myException.Errors(i).Source + ControlChars.Cr _
               + "SQL: " + myException.Errors(i).SQLState + ControlChars.Cr)
        Next i
    End Try
End Sub

[C#]
public void ThrowIfxException()
{
   string mySelectQuery = "SELECT column1 FROM table1";
   IfxConnection myConnection =
      new IfxConnection("DATABASE=BADDB;");
   IfxCommand myCommand = new IfxCommand(mySelectQuery,myConnection);
   try
   {
      myCommand.Connection.Open();
   }
   catch (IfxException myException)
   {
      for (int i=0; i < myException.Errors.Count; i++)
     {
         MessageBox.Show("Index #" + i + "\n" +
             "Message: " + myException.Errors[i].Message + "\n" +
             "Native: " + myException.Errors[i].NativeError.ToString() + "\n" +
             "Source: " + myException.Errors[i].Source + "\n" +
             "SQL: " + myException.Errors[i].SQLState + "\n");
     }
   }
}
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxException Members" on page 5-405

"IBM.Data.Informix Namespace" on page 5-1

"IfxError Class" on page 5-384
"IfxErrorCollection Class" on page 5-395

## IfxException Members

IfxException overview

### Public Properties

| Property | Description |
|---|---|
| Errors | Gets a collection of one or more IfxError objects that give detailed information about exceptions generated by the IBM Data Server Provider for .NET. |
| HelpLink (inherited from Exception) | Gets or sets a link to the help file associated with this exception. |
| InnerException (inherited from Exception) | Gets the Exception instance that caused the current exception. |
| Message | Overridden. Gets the text describing the error. |
| Source | Overridden. Gets the name of the driver that generated the error. |
| StackTrace (inherited from Exception) | Gets a string representation of the frames on the call stack at the time the current exception was thrown. |
| TargetSite (inherited from Exception) | Gets the method that throws the current exception. |

### Public Methods

| Method | Description |
|---|---|
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetBaseException (inherited from Exception) | When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetObjectData | Overridden. See Exception.GetObjectData. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| ToString (inherited from Exception) | Overridden. Creates and returns a string representation of the current exception. |

### Protected Properties

| Property | Description |
|---|---|
|  HResult (inherited from Exception) | Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. |

### Protected Methods

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxException Class" on page 5-403

"IBM.Data.Informix Namespace" on page 5-1

## IfxException Methods

The methods of the IfxException class are listed here. For a complete list of IfxException class members, see the IfxException Members topic.

### Public Methods

| Method | Description |
|---|---|
|  Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
|  GetBaseException (inherited from Exception) | When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
|  GetObjectData | Overridden. See Exception.GetObjectData. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  ToString (inherited from Exception) | Overridden. Creates and returns a string representation of the current exception. |

## Protected Methods

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxException Class" on page 5-403

"IBM.Data.Informix Namespace" on page 5-1

**IfxException.GetObjectData Method:**

This member overrides System.Exception.GetObjectData.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Overrides Public Sub GetObjectData( _
   ByVal si As SerializationInfo, _
   ByVal context As StreamingContext _
) Implements ISerializable.GetObjectData
[C#]
[Serializable]
public override void GetObjectData(
   SerializationInfo si,
   StreamingContext context
);
[C++]
[Serializable]
public: void GetObjectData(
   SerializationInfo* si,
   StreamingContext context
);
[JScript]
public
   Serializable
override function GetObjectData(
   si : SerializationInfo,
   context : StreamingContext
);
```

**Reference**

"IfxException Class" on page 5-403

"IfxException Members" on page 5-405

"IBM.Data.Informix Namespace" on page 5-1

## IfxException Properties

The properties of the IfxException class are listed here. For a complete list of IfxException class members, see the IfxException Members topic.

### Public Properties

| Property | Description |
|---|---|
| Errors | Gets a collection of one or more IfxError objects that give detailed information about exceptions generated by the IBM Data Server Provider for .NET. |
| HelpLink (inherited from Exception) | Gets or sets a link to the help file associated with this exception. |
| InnerException (inherited from Exception) | Gets the Exception instance that caused the current exception. |
| Message | Overridden. Gets the text describing the error. |
| Source | Overridden. Gets the name of the driver that generated the error. |
| StackTrace (inherited from Exception) | Gets a string representation of the frames on the call stack at the time the current exception was thrown. |
| TargetSite (inherited from Exception) | Gets the method that throws the current exception. |

### Protected Properties

| Property | Description |
|---|---|
| HResult (inherited from Exception) | Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. |

**Reference**

"IfxException Class" on page 5-403

"IBM.Data.Informix Namespace" on page 5-1

**IfxException.Errors Property:**

Gets a collection of one or more IfxError objects that give detailed information about exceptions generated by the IBM Data Server Provider for .NET.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Public ReadOnly Property Errors As IfxErrorCollection

[C#]
[Serializable]
```

```
public IfxErrorCollection
 Errors {get;}
[C++]
[Serializable]
public: __property IfxErrorCollection
* get_Errors();
[JScript]
public
   Serializable
function get Errors() : IfxErrorCollection
;
```

**Property value**

The collected instances of the IfxError class.

**Remarks**

This property is a wrapper for the IfxErrorCollection .

**Example**

[Visual Basic, C#] The following example displays each IfxError within a IfxErrorCollection collection.

```
[Visual Basic]
Public Sub DisplayIfxErrorCollection(myException As IfxException)
    Dim i As Integer
    For i = 0 To myException.Errors.Count - 1
        MessageBox.Show("Index #" + i.ToString() + ControlChars.Cr _
           + "Message: " + myException.Errors(i).Message + ControlChars.Cr _
           + "Native: " + myException.Errors(i).NativeError.ToString()
                        + ControlChars.Cr _
           + "Source: " + myException.Errors(i).Source + ControlChars.Cr _
           + "SQL: " + myException.Errors(i).SQLState + ControlChars.Cr)
    Next i
End Sub

[C#]
public void DisplayIfxErrorCollection(IfxException myException)
{
   for (int i=0; i < myException.Errors.Count; i++)
   {
      MessageBox.Show("Index #" + i + "\n" +
             "Message: " + myException.Errors[i].Message + "\n" +
             "Native: " + myException.Errors[i].NativeError.ToString() + "\n" +
             "Source: " + myException.Errors[i].Source + "\n" +
             "SQL: " + myException.Errors[i].SQLState + "\n");
   }
}
```

**Reference**

"IfxException Class" on page 5-403

"IfxException Members" on page 5-405

"IBM.Data.Informix Namespace" on page 5-1

**IfxException.Message Property:**

Gets the text describing the error.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Overrides Public ReadOnly Property Message As String
[C#]
[Serializable]
public override string Message {get;}
[C++]
[Serializable]
public: __property virtual String* get_Message();
[JScript]
public
   Serializable
function get Message() : String;
```

**Property value**

The text describing the error.

**Remarks**

This is a wrapper for the IfxError.Message property of the first IfxError in the Errors property.

**Example**

[Visual Basic, C#] The following example displays the IfxError.Message of the first IfxError within a IfxErrorCollection collection.

```
[Visual Basic]
Public Sub DisplayIfxErrorCollection(myException As IfxException)
    MessageBox.Show("Message: " + myException.Message + ControlChars.Cr _
       + "StackTrace: " + myException.StackTrace + ControlChars.Cr
End Sub
End Class

[C#]
public void DisplayIfxErrorCollection(IfxException myException)
{
   MessageBox.Show("Message: " + myException.Message + "\n" +
          "StackTrace: " + myException.StackTrace + "\n";
}
```

**Reference**

"IfxException Class" on page 5-403

"IfxException Members" on page 5-405

"IBM.Data.Informix Namespace" on page 5-1

**IfxException.Source Property:**

Gets the name of the driver that generated the error.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Overrides Public ReadOnly Property Source As String
[C#]
[Serializable]
public override string Source {get;}
[C++]
[Serializable]
public: __property virtual String* get_Source();
[JScript]
public
   Serializable
function get Source() : String;
```

**Property value**

IBM.Data.DB2

**Remarks**

This is a wrapper for the IfxError.Source property of the first IfxError in the Errors collection.

**Example**

[Visual Basic, C#] The following example displays the IfxError.Message, IfxError.Source, and stack trace properties of the first IfxError within the IfxErrorCollection collection.

```
[Visual Basic]
Public Sub DisplayIfxErrorCollection(myException As IfxException)
    MessageBox.Show("Message: " + myException.Message + ControlChars.Cr _
       + "StackTrace: " + myException.StackTrace + ControlChars.Cr _
       + "Source: " + myException.Source + ControlChars.Cr)
End Sub

[C#]
public void DisplayIfxErrorCollection(IfxException myException) {
   MessageBox.Show("Message: " + myException.Message + "\n" +
          "StackTrace: " + myException.StackTrace + "\n" +
          "Source: " + myException.Source + "\n");
}
```

**Reference**

"IfxException Class" on page 5-403

"IfxException Members" on page 5-405

"IBM.Data.Informix Namespace" on page 5-1

## IfxFactory Class

Represents a set of methods for creating instances of the System.Data.Common data source classes for the IBM Data Server Provider for .NET.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
    System.Data.Common.DbProviderFactory
        IBM.Data.Informix.IfxFactory

## Syntax

```
[Visual Basic]
Public NotInheritable Class IfxFactory
 Inherits DbProviderFactory
[C#]
public sealed class IfxFactory : DbProviderFactory
[C++]
public ref class IfxFactory sealed : public DbProviderFactory
[JScript]
public final class IfxFactory extends DbProviderFactory
```

## Example

[C#] The following example demonstrates the creation of a DbConnection instance, using the IfxFactory.CreateConnection method.

```
[C#]
IfxFactory myFactory = IfxFactory.Instance;
DbConnection conn = myFactory.CreateConnection();
```

[C#] You can also get a database factory instance by using the generic DbProviderFactories class. In the following example, which accomplishes the same task as the previous example, note the `"IBM.Data.Informix"` string passed to the GetFactory method. This string uniquely identifies the IBM Data Server Provider for .NET from all other .NET data providers.

```
[C#]
DbProviderFactory myFactory = DbProviderFactories.GetFactory("IBM.Data.Informix");
DbConnection conn = myFactory.CreateConnection();
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0 and 3.0

**Reference**

"IfxFactory Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxFactory Members

The members of the IfxFactory class are listed here.

### Public Properties

| Name | Description |
|---|---|
|  CanCreateDataSourceEnumerator | Overridden. Gets a boolean value indicating whether the IfxFactory can create a DbDataSourceEnumerator instance. |

### Public Methods

| Name | Description |
|---|---|
| CreateCommand | Overridden. Returns a DbCommand instance. |
| CreateCommandBuilder | Overridden. Returns a DbCommandBuilder instance. |
| CreateConnection | Overridden. Returns a DbConnection instance. |
| CreateConnectionStringBuilder | Overridden. Returns a DbConnectionStringBuilder instance. |
| CreateDataAdapter | Overridden. Returns a DbDataAdapter instance. |
| CreateDataSourceEnumerator | Overridden. Returns a DbDataSourceEnumerator instance. |
| CreateParameter | Overridden. Returns a DbParameter instance. |
| CreatePermission | Overridden. Returns a System.Security.CodeAccessPermission instance. |

### Public Fields

| Name | Description |
|---|---|
| Instance | Gets a IfxFactory instance. |

**Reference**

"IfxFactory Class" on page 5-411
"IBM.Data.Informix Namespace" on page 5-1

## IfxFactory Fields

The fields of the IfxFactory class are listed here.

### Public Fields

| Name | Description |
|---|---|
| Instance | Gets a IfxFactory instance. |

**Reference**

"IfxFactory Class" on page 5-411
"IfxFactory Members" on page 5-412
"IBM.Data.Informix Namespace" on page 5-1

**IfxFactory.Instance Field:**

Gets a IfxFactory instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Instance As IfxFactory
[C#]
public static readonly IfxFactory Instance
[C++]
public:
static initonly IfxFactory^ Instance
[JScript]
public static final var Instance () : IfxFactory
```

**Example**

[C#] The following example demonstrates the creation of a IfxFactory instance.

```
[C#]
    IfxFactory myFactory = IfxFactory.Instance;
```

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxFactory Class" on page 5-411
"IfxFactory Members" on page 5-412
"IBM.Data.Informix Namespace" on page 5-1

## IfxFactory Methods

The methods of the IfxFactory class are listed here.

### Public Methods

| Name | Description |
|---|---|
|  CreateCommand | Overridden. Returns a DbCommand instance. |
|  CreateCommandBuilder | Overridden. Returns a DbCommandBuilder instance. |
|  CreateConnection | Overridden. Returns a DbConnection instance. |
|  CreateConnectionStringBuilder | Overridden. Returns a DbConnectionStringBuilder instance. |
|  CreateDataAdapter | Overridden. Returns a DbDataAdapter instance. |
|  CreateDataSourceEnumerator | Overridden. Returns a DbDataSourceEnumerator instance. |
|  CreateParameter | Overridden. Returns a DbParameter instance. |
|  CreatePermission | Overridden. Returns a System.Security.CodeAccessPermission instance. |

**Reference**

"IfxFactory Class" on page 5-411

"IfxFactory Members" on page 5-412

"IBM.Data.Informix Namespace" on page 5-1

**IfxFactory.CreateCommandBuilder Method:**

Returns a DbCommandBuilder instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function CreateCommandBuilder As DbCommandBuilder
[C#]
public override DbCommandBuilder CreateCommandBuilder ()
[C++]
public:
virtual DbCommandBuilder^ CreateCommandBuilder () override
[JScript]
public override function CreateCommandBuilder () : DbCommandBuilder
```

**Return value**

A DbCommandBuilder instance, which will enable data exchange with DB2 family databases, through the IBM Data Server Provider for .NET.

**Example**

[C#] The following example demonstrates the creation of a DbCommandBuilder instance, using the IfxFactory.CreateCommandBuilder method.

```
[C#]
    IfxFactory myFactory = IfxFactory.Instance;
    DbCommandBuilder cmdbld = myFactory.CreateCommandBuilder();
```

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxFactory Class" on page 5-411

"IfxFactory Members" on page 5-412

"IBM.Data.Informix Namespace" on page 5-1

**IfxFactory.CreateCommand Method:**

Returns a DbCommand instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function CreateCommand As DbCommand
[C#]
public override DbCommand CreateCommand ()
[C++]
public:
virtual DbCommand^ CreateCommand () override
[JScript]
public override function CreateCommand () : DbCommand
```

**Return value**

A DbCommand instance, which can represent SQL statements or stored procedures you run against DB2 family databases through the IBM Data Server Provider for .NET.

**Example**

[C#] The following example demonstrates the creation of a DbCommand instance, using the IfxFactory.CreateCommand method.

```
[C#]
    IfxFactory myFactory = IfxFactory.Instance;
    DbCommand cmd = myFactory.CreateCommand();
```

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxFactory Class" on page 5-411

"IfxFactory Members" on page 5-412

"IBM.Data.Informix Namespace" on page 5-1

**IfxFactory.CreateConnection Method:**

Returns a DbConnection instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function CreateConnection As DbConnection
[C#]
public override DbConnection CreateConnection ()
[C++]
public:
virtual DbConnection^ CreateConnection () override
[JScript]
public override function CreateConnection () : DbConnection
```

**Return value**

A DbConnection instance, which will enable connections to DB2 family databases through the IBM Data Server Provider for .NET.

### Example

[C#] The following example demonstrates the creation of a DbConnection instance, using the IfxFactory.CreateConnection method.

```
[C#]
    IfxFactory myFactory = IfxFactory.Instance;
    DbConnection conn = myFactory.CreateConnection();
```

### Version information

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxFactory Class" on page 5-411

"IfxFactory Members" on page 5-412

"IBM.Data.Informix Namespace" on page 5-1

## IfxFactory.CreateConnectionStringBuilder Method:

Returns a DbConnectionStringBuilder instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Overrides Function CreateConnectionStringBuilder As DbConnectionStringBuilder
[C#]
public override DbConnectionStringBuilder CreateConnectionStringBuilder ()
[C++]
public:
virtual DbConnectionStringBuilder^ CreateConnectionStringBuilder () override
[JScript]
public override function CreateConnectionStringBuilder () :
  DbConnectionStringBuilder
```

### Return value

A DbConnectionStringBuilder instance, which will facilitate the creation of connection strings for an IBM data server.

### Example

[C#] The following example demonstrates the creation of a DbConnectionStringBuilder instance, using the IfxFactory.ConnectionStringBuilder method.

```
[C#]
    IfxFactory myFactory = IfxFactory.Instance;
    DbConnectionStringBuilder stringbuilder =
    myFactory.CreateConnectionStringBuilder();
```

### Version information

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxFactory Class" on page 5-411
"IfxFactory Members" on page 5-412
"IBM.Data.Informix Namespace" on page 5-1

**IfxFactory.CreateDataAdapter Method:**

Returns a DbDataAdapter instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function CreateDataAdapter As DbDataAdapter
[C#]
public override DbDataAdapter CreateDataAdapter ()
[C++]
public:
virtual DbDataAdapter^ CreateDataAdapter () override
[JScript]
public override function CreateDataAdapter () : DbDataAdapter
```

**Return value**

A DbDataAdapter instance, which will enable data exchange with DB2 family databases, through the IBM Data Server Provider for .NET.

**Example**

[C#] The following example demonstrates the creation of a DbDataAdapter instance, using the IfxFactory.CreateDataAdapter method.

```
[C#]
    IfxFactory myFactory = IfxFactory.Instance;
    DbDataAdapter adapter = myFactory.CreateDataAdapter();
```

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxFactory Class" on page 5-411
"IfxFactory Members" on page 5-412
"IBM.Data.Informix Namespace" on page 5-1

**IfxFactory.CreateDataSourceEnumerator Method:**

Returns a DbDataSourceEnumerator instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function CreateDataSourceEnumerator As DbDataSourceEnumerator
[C#]
public override DbDataSourceEnumerator CreateDataSourceEnumerator ()
[C++]
public:
virtual DbDataSourceEnumerator^ CreateDataSourceEnumerator () override
[JScript]
public override function CreateDataSourceEnumerator () : DbDataSourceEnumerator
```

**Return value**

A DbDataSourceEnumerator instance, which will enable you to find visible DB2 family database server instances and databases on those instances. The DbDataSourceEnumerator instance works through the IBM Data Server Provider for .NET and is able to see the following:

- all databases cataloged on the local instance. In the case where there are multiple running local instances, only the instance to which the application is attached will be visible.
- all visible remote database server instances and their databases. In order to be visible, the remote database server instances must have a running database administration server (DAS), which is configured to respond to Search and Known Discovery requests.
-

Database server instances without any databases will be visible.

**Example**

[C#] The following example demonstrates the creation of a DbDataSourceEnumerator instance, using the IfxFactory.CreateDataSourceEnumerator method.

```
[C#]
    IfxFactory myFactory = IfxFactory.Instance;
    if (myFactory.CanCreateDataSourceEnumerator)
    {
        DbDataSourceEnumerator dsenum =
            myFactory.CreateDataSourceEnumerator();
    }
```

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxFactory Class" on page 5-411

"IfxFactory Members" on page 5-412

"IBM.Data.Informix Namespace" on page 5-1

**IfxFactory.CreateParameter Method:**

Returns a DbParameter instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function CreateParameter As DbParameter
[C#]
public override DbParameter CreateParameter ()
[C++]
public:
virtual DbParameter^ CreateParameter () override
[JScript]
public override function CreateParameter () : DbParameter
```

**Return value**

A DbParameter instance, which can represent parameters to a DbCommand instance.

**Example**

[C#] The following example demonstrates the creation of a DbParameter instance, using the IfxFactory.CreateParameter method.

```
[C#]
    IfxFactory myFactory = IfxFactory.Instance;
    DbParameter param = myFactory.CreateParameter();
```

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxFactory Class" on page 5-411

"IfxFactory Members" on page 5-412

"IBM.Data.Informix Namespace" on page 5-1

**IfxFactory.CreatePermission Method (System.Security.Permissions.PermissionState):**

Returns a System.Security.CodeAccessPermission instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function CreatePermission ( _
   state As PermissionState _
) As CodeAccessPermission
[C#]
public override CodeAccessPermission CreatePermission (
   PermissionState state
)
[C++]
public:
virtual CodeAccessPermission^ CreatePermission (
   PermissionState state
```

```
) override
[JScript]
public override function CreatePermission (
   state: PermissionState
) : CodeAccessPermission
```

**Parameters**

**state** An instance of System.Security.Permissions.PermissionState.

**Return value**

A System.Security.CodeAccessPermission instance.

**Example**

[C#] The following example demonstrates the creation of a CodeAccessPermission instance, using the IfxFactory.CreatePermission method.

```
[C#]
    IfxFactory myFactory = IfxFactory.Instance;
    System.Security.Permissions.PermissionState state =
        System.Security.Permissions.PermissionState.None;
    CodeAccessPermission permission = myFactory.CreatePermission(state);
```

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxFactory Class" on page 5-411

"IfxFactory Members" on page 5-412

"IBM.Data.Informix Namespace" on page 5-1

## IfxFactory Properties

The properties of the IfxFactory class are listed here.

### Public Properties

| Name | Description |
|---|---|
|  CanCreateDataSourceEnumerator | Overridden. Gets a boolean value indicating whether the IfxFactory can create a DbDataSourceEnumerator instance. |

**Reference**

"IfxFactory Class" on page 5-411

"IfxFactory Members" on page 5-412

"IBM.Data.Informix Namespace" on page 5-1

**IfxFactory.CanCreateDataSourceEnumerator Property:**

Gets a boolean value indicating whether the IfxFactory can create a DbDataSourceEnumerator instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides ReadOnly Property CanCreateDataSourceEnumerator As Boolean
[C#]
public override bool CanCreateDataSourceEnumerator { get; }
[C++]
public:
virtual property bool CanCreateDataSourceEnumerator {
   bool get () override;
}
[JScript]
public override function get CanCreateDataSourceEnumerator () : boolean
```

**Property value**

`true` if the IfxFactory can create a DbDataSourceEnumerator instance; otherwise `false`.

**Example**

[C#] The following example demonstrates the use of the CanCreateDataSourceEnumerator property, in the creation of a DbDataSourceEnumerator instance.

```
[C#]
    IfxFactory myFactory = IfxFactory.Instance;
    if (myFactory.CanCreateDataSourceEnumerator)
    {
        DbDataSourceEnumerator dsenum =
            myFactory.CreateDataSourceEnumerator();
    }
```

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxFactory Class" on page 5-411

"IfxFactory.CreateDataSourceEnumerator Method" on page 5-418

"IfxFactory Members" on page 5-412

"IBM.Data.Informix Namespace" on page 5-1

# IfxFormatException Class

The exception that is generated when the Parse method for the applicable IBM.Data.Informix class objects includes a string with non-numeric values.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
  System.Exception

System.SystemException
IBM.Data.Informix.IfxTypeException
IBM.Data.Informix.IfxFormatException

## Syntax

```
[Visual Basic]
Public Class IfxFormatException
[C#]
public class IfxFormatException
[C++]
public __gc class IfxFormatException
[JScript]
public class IfxFormatException
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxFormatException Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxFormatException Members

The exception that is generated when the Parse method for the applicable IBM.Data.Informix structure or class includes a string with non-numeric values. The following tables list the members exposed by the IfxFormatException class.

### Public Properties

| Property | Description |
|---|---|
| HelpLink (inherited from Exception) | Gets or sets a link to the help file associated with this exception. |
| InnerException (inherited from Exception) | Gets the Exception instance that caused the current exception. |
| Message (inherited from Exception | Gets the text describing the error. |
| Source (inherited from Exception | Gets the name of the application or the object that generated the error. |
| StackTrace (inherited from Exception) | Gets a string representation of the frames on the call stack at the time the current exception was thrown. |
| TargetSite (inherited from Exception) | Gets the method that throws the current exception. |

### Public Methods

| Method | Description |
|---|---|
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetBaseException (inherited from Exception) | When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetObjectData (inherited from Object) | Overridden. See Exception.GetObjectData. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| ToString (inherited from Exception) | Overridden. Creates and returns a string representation of the current exception. |

### Protected Properties

| Property | Description |
|---|---|
| HResult (inherited from Exception) | Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. |

### Protected Methods

| Method | Description |
|---|---|
| Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxFormatException Class" on page 5-422

"IBM.Data.Informix Namespace" on page 5-1

## IfxInfoMessageEventArgs Class

Provides data for the InfoMessage event..

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
    System.EventArgs
        IBM.Data.Informix.IfxInfoMessageEventArgs

## Syntax

```
[Visual Basic]
NotInheritable Public Class IfxInfoMessageEventArgs
   Inherits EventArgs
[C#]
public sealed class IfxInfoMessageEventArgs : EventArgs
[C++]
public __gc __sealed class IfxInfoMessageEventArgs : public
   EventArgs
[JScript]
public class IfxInfoMessageEventArgs extends EventArgs
```

## Remarks

The InfoMessage event contains a IfxErrorCollection collection with warnings sent from the database.

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInfoMessageEventArgs Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxInfoMessageEventArgs Members

IfxInfoMessageEventArgs overview

### Public Properties

| Property | Description |
|---|---|
|  Errors | Gets the collection of warnings sent from the database. |

### Public Methods

| Method | Description |
|---|---|
|  Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |

| Method | Description |
|---|---|
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  ToString (inherited from Object) | Returns a String that represents the current Object. |

### Protected Methods

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxInfoMessageEventArgs Class" on page 5-424

"IBM.Data.Informix Namespace" on page 5-1

## IfxInfoMessageEventArgs Properties

The properties of the IfxInfoMessageEventArgs class are listed here. For a complete list of IfxInfoMessageEventArgs class members, see the IfxInfoMessageEventArgs Members topic.

### Public Properties

| Property | Description |
|---|---|
|  Errors | Gets the collection of warnings sent from the database. |

**Reference**

"IfxInfoMessageEventArgs Class" on page 5-424

"IBM.Data.Informix Namespace" on page 5-1

**IfxInfoMessageEventArgs.Errors Property:**

Gets the collection of warnings sent from the database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Errors As IfxErrorCollection
```

```
[C#]
public IfxErrorCollection
 Errors {get;}
[C++]
public: __property IfxErrorCollection
* get_Errors();
[JScript]
public function get Errors() : IfxErrorCollection
;
```

**Property value**

The collection of warnings sent from the database.

**Reference**

"IfxInfoMessageEventArgs Class" on page 5-424

"IfxInfoMessageEventArgs Members" on page 5-425

"IBM.Data.Informix Namespace" on page 5-1

**IfxInfoMessageEventHandler Delegate:**

Represents the method that will handle the InfoMessage event of a IfxConnection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Public Delegate Sub IfxInfoMessageEventHandler( _
   ByVal sender As Object, _
   ByVal e As IfxInfoMessageEventArgs _
)
[C#]
[Serializable]
public delegate void IfxInfoMessageEventHandler(
   object sender,
   IfxInfoMessageEventArgs e
);
[C++]
[Serializable]
public __gc __delegate void IfxInfoMessageEventHandler(
   Object* sender,
   IfxInfoMessageEventArgs* e
);
```

**Remarks**

When you create a IfxInfoMessageEventArgs delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see "Events and Delegates" in the .NET Framework SDK documentation.

**Reference**

"IBM.Data.Informix Namespace" on page 5-1

# IfxInt16 Structure

Represents the SMALLINT Informix data type. Encapsulates the short .NET data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
Public Structure IfxInt16
[C#]
public struct IfxInt16
[C++]
public value class IfxInt16
```

## Example

[C#] The following example demonstrates how to retrieve a single SMALLINT column value from a table.

```
[C#]
  public static string getParam(IfxConnection conn)
  {
    string mySelectQuery = "SELECT * FROM EMPLOYEE";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);
    IfxDataReader reader = myCommand.ExecuteReader();

    if (reader.Read())
    {
      IfxInt16 selectValue = reader.GetIfxInt16(8);

      if (!selectValue.IsNull) { return selectValue.ToString(); }
    }

    return "NULL";
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt16 Members

Represents the SMALLINT Informix data type. Encapsulates the short .NET data type. The following tables list the members exposed by the IfxInt16 class.

### Public Fields

| Field | Description |
|---|---|
| MaxValue | Represents the maximum value for IfxInt16: 32,767. |
| MinValue | Represents the minumum value for IfxInt16: -32,768. |
| Null | Represents the null value for IfxInt16. |
| Zero | Represents the zero value for IfxInt16. |

### Public Constructors

| Constructor | Description |
|---|---|
| IfxInt16 | Initializes a new IfxInt16 object with the specified value. |

### Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxInt16 structure is null. |
| Value | Gets the value stored in the IfxInt16 structure. |

### Public Methods

| Method | Description |
|---|---|
| Parse | Converts the supplied String to IfxInt16. |
| ToString | Returns a string that represents the IfxInt16 structure. |
| op_explicit | Converts the supplied IfxInt16 structure to a short integer. |
| op_implicit | Converts the supplied short integer to IfxInt16. |

**Reference**

"IfxInt16 Structure" on page 5-428

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt16 Fields

The fields of the IfxInt16 structure are listed here.

## Public Fields

| Field | Description |
|---|---|
| MaxValue | Represents the maximum value for IfxInt16: 32,767. |
| MinValue | Represents the minumum value for IfxInt16: -32,768. |
| Null | Represents the null value for IfxInt16. |
| Zero | Represents the zero value for IfxInt16. |

**Reference**

"IfxInt16 Members" on page 5-428
"IfxInt16 Structure" on page 5-428
"IBM.Data.Informix Namespace" on page 5-1

### IfxInt16.MaxValue Field:

Represents the maximum value for IfxInt16: 32,767.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MaxValue As IfxInt16
[C#]
public static readonly IfxInt16 MaxValue
[C++]
public:
static initonly IfxInt16 MaxValue
[JScript]
public static final var MaxValue () : IfxInt16
```

**Remarks**

The value of this constant is `32,767`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Structure" on page 5-428
"IBM.Data.Informix Namespace" on page 5-1

### IfxInt16.MinValue Field:

Represents the minumum value for IfxInt16: -32,768.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MinValue As IfxInt16
[C#]
public static readonly IfxInt16 MinValue
[C++]
public:
static initonly IfxInt16 MinValue
[JScript]
public static final var MinValue () : IfxInt16
```

**Remarks**

The value of this constant is `-32,768`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Structure" on page 5-428

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt16.Null Field:**

Represents the null value for IfxInt16.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxInt16
[C#]
public static readonly IfxInt16 Null
[C++]
public:
static initonly IfxInt16 Null
[JScript]
public static final var Null () : IfxInt16
```

**Remarks**

The value of this constant is `NULL`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Structure" on page 5-428

"IBM.Data.Informix Namespace" on page 5-1

### IfxInt16.Zero Field:

Represents the zero value for IfxInt16.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

#### Syntax

```
[Visual Basic]
Public Shared ReadOnly Zero As IfxInt16
[C#]
public static readonly IfxInt16 Zero
[C++]
public:
static initonly IfxInt16 Zero
[JScript]
public static final var Zero () : IfxInt16
```

#### Remarks

The value of this constant is `0`.

#### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Structure" on page 5-428

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt16 Constructor

Initializes a new IfxInt16 structure with the specified value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New(value as Short)
[C#]
public IfxInt16(short value);
[C++]
public: IfxInt16(short value);
[JScript]
public function IfxInt16(value : short);
```

### Parameters

**value** A short integer value to populate the IfxInt16 instance.

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Structure" on page 5-428
"IBM.Data.Informix Namespace" on page 5-1

## IfxInt16 Methods

The methods of the IfxInt16 structure are listed here.

### Public Methods

| Method | Description |
|---|---|
| Parse | Converts the supplied String to IfxInt16. |
| ToString | Returns a string that represents the IfxInt16 structure. |
| op_explicit | Converts the supplied IfxInt16 structure to a short integer. |
| op_implicit | Converts the supplied short integer to IfxInt16. |

**Reference**

"IfxInt16 Members" on page 5-428
"IfxInt16 Structure" on page 5-428
"IBM.Data.Informix Namespace" on page 5-1

**IfxInt16.op_explicit Method:**

Converts the supplied IfxInt16 structure to a short integer.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Narrowing Operator CType (source As IfxInt16) As Short
[C#]
public static implicit operator short (IfxInt16 source)
[C++]
public:
static implicit operator short (IfxInt16 source)
```

**Parameters**

**source** A IfxInt16 structure to be converted to a short integer.

**Return value**

A short integer value converted from a IfxInt16 instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Structure" on page 5-428

"IBM.Data.Informix Namespace" on page 5-1

### IfxInt16.op_implicit Method:

Converts the supplied short integer to IfxInt16.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

#### Syntax

```
[Visual Basic]
Public Shared Widening Operator CType (source As Short) As IfxInt16
[C#]
public static implicit operator IfxInt16 (short source)
[C++]
public:
static implicit operator IfxInt16 (short source)
```

#### Parameters

**source** A short integer value to be converted to IfxInt16.

#### Return value

A IfxInt16 structure with the value of **source**.

#### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Structure" on page 5-428
"IBM.Data.Informix Namespace" on page 5-1

### IfxInt16.Parse Method:

Converts the supplied String to IfxInt16.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

#### Syntax

```
[Visual Basic]
Public Shared Function Parse (szInt16 As String) As IfxInt16
[C#]
public static IfxInt16 Parse (string szInt16)
[C++]
public:
static IfxInt16 Parse (string szInt16)
[JScript]
public static function Parse (szInt16 String ) : IfxInt16
```

**Parameters**

**szInt16**
A String value to be converted to IfxInt16.

**Return value**

A IfxInt16 structure with the numeric value of **szInt16**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Structure" on page 5-428

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt16.ToString Method:**

Returns a string that represents the IfxInt16 structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxInt16 structure.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Structure" on page 5-428

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt16 Properties

The properties of the IfxInt16 structure are listed here.

### Public Properties

| Property | Description |
|---|---|
|  IsNull | Gets a value that indicates if the value stored in the IfxInt16 structure is null. |

| Property | Description |
|---|---|
|  Value | Gets the value stored in the IfxInt16 structure. |

**Reference**

"IfxInt16 Members" on page 5-428

"IfxInt16 Structure" on page 5-428

"IBM.Data.Informix Namespace" on page 5-1

### IfxInt16.IsNull Property:

Gets a value that indicates if the value stored in the IfxInt16 structure is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Structure" on page 5-428

"IBM.Data.Informix Namespace" on page 5-1

### IfxInt16.Value Property:

Gets the value stored in the IfxInt16 structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Value As Short
[C#]
public short Value {get;}
```

```
[C++]
public: __property short get_Value();
[JScript]
public function get Value() : short;
```

**Property value**

A short integer representing the IfxInt16 instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Structure" on page 5-428

"IBM.Data.Informix Namespace" on page 5-1

# IfxInt32 Structure

Represents the INTEGER Informix data type. Encapsulates the int .NET data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
Public Structure IfxInt32
[C#]
public struct IfxInt32
[C++]
public value class IfxInt32
```

## Example

[C#] The following example demonstrates how to retrieve a single INT column value from a table.

```
[C#]
  public static string getParam(IfxConnection conn)
  {
    string mySelectQuery = "SELECT * FROM SALES";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);
    IfxDataReader reader = myCommand.ExecuteReader();

    if (reader.Read())
    {
      IfxInt32 selectValue = reader.GetIfxInt32(3);

      if (!selectValue.IsNull) { return selectValue.ToString(); }
    }

    return "NULL";
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt32 Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt32 Members

Represents the INTEGER Informix data type. Encapsulates the int .NET data type. The following tables list the members exposed by the IfxInt32 class.

### Public Fields

| Field | Description |
|---|---|
| MaxValue | Represents the maximum value for IfxInt32: 2,147,483,647. |
| MinValue | Represents the minumum value for IfxInt32: -2,147,483,648. |
| Null | Represents the null value for IfxInt32. |
| Zero | Represents the zero value for IfxInt32. |

### Public Constructors

| Constructor | Description |
|---|---|
| IfxInt32 | Initializes a new IfxInt32 structure with the specified value. |

### Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxInt32 structure is null. |
| Value | Gets the value stored in the IfxInt32 structure. |

### Public Methods

| Method | Description |
|---|---|
| Parse | Converts the supplied String to IfxInt32. |
| ToString | Returns a string that represents the IfxInt32 structure. |
| op_explicit | Converts the supplied IfxInt32 structure to an integer. |
| op_implicit | Converts the supplied integer to IfxInt32. |

**Reference**

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt32 Fields

The fields of the IfxInt32 structure are listed here.

### Public Fields

| Field | Description |
|---|---|
| MaxValue | Represents the maximum value for IfxInt32: 2,147,483,647. |
| MinValue | Represents the minumum value for IfxInt32: -2,147,483,648. |
| Null | Represents the null value for IfxInt32. |
| Zero | Represents the zero value for IfxInt32. |

**Reference**

"IfxInt32 Members" on page 5-438

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt32.MaxValue Field:**

Represents the maximum value for IfxInt32: 2,147,483,647.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MaxValue As IfxInt32
[C#]
public static readonly IfxInt32 MaxValue
[C++]
public:
static initonly IfxInt32 MaxValue
[JScript]
public static final var MaxValue () : IfxInt32
```

**Remarks**

The value of this constant is `2,147,483,647`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt32.MinValue Field:**

Represents the minumum value for IfxInt32: -2,147,483,648.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MinValue As IfxInt32
[C#]
public static readonly IfxInt32 MinValue
[C++]
public:
static initonly IfxInt32 MinValue
[JScript]
public static final var MinValue () : IfxInt32
```

**Remarks**

The value of this constant is `-2,147,483,648`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt32 Structure" on page 5-437
"IBM.Data.Informix Namespace" on page 5-1

**IfxInt32.Null Field:**

Represents the null value for IfxInt32.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxInt32
[C#]
public static readonly IfxInt32 Null
[C++]
public:
static initonly IfxInt32 Null
[JScript]
public static final var Null () : IfxInt32
```

**Remarks**

The value of this constant is `NULL`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt32.Zero Field:**

Represents the zero value for IfxInt32.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Zero As IfxInt32
[C#]
public static readonly IfxInt32 Zero
[C++]
public:
static initonly IfxInt32 Zero
[JScript]
public static final var Zero () : IfxInt32
```

**Remarks**

The value of this constant is `0`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt32 Constructor

Initializes a new IfxInt32 structure with the specified value.

### Syntax

```
[Visual Basic]
Public Sub New(value as Integer)
[C#]
public IfxInt32(int value);
[C++]
public: IfxInt32(int value);
[JScript]
public function IfxInt32(value : int);
```

### Parameters

**value** An integer value to populate the IfxInt32 instance.

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt32 Methods

The methods of the IfxInt32 structure are listed here.

### Public Methods

| Method | Description |
|---|---|
|  Parse | Converts the supplied String to IfxInt32. |
|  ToString | Returns a string that represents the IfxInt32 structure. |
|  op_explicit | Converts the supplied IfxInt32 structure to an integer. |
|  op_implicit | Converts the supplied integer to IfxInt32. |

**Reference**

"IfxInt32 Members" on page 5-438

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt32.op_explicit Method:**

Converts the supplied IfxInt32 structure to an integer.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Narrowing Operator CType (source As IfxInt32) As Integer
[C#]
public static implicit operator int (IfxInt32 source)
[C++]
public:
static implicit operator int (IfxInt32 source)
```

**Parameters**

**source** A IfxInt32 structure to be converted to an integer.

**Return value**

An integer value converted from a IfxInt32 instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt32.op_implicit Method:**

Converts the supplied integer to IfxInt32.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Widening Operator CType (source As Integer) As IfxInt32
[C#]
public static implicit operator IfxInt32 (int source)
[C++]
public:
static implicit operator IfxInt32 (int source)
```

**Parameters**

**source** An integer value to be converted to IfxInt32.

**Return value**

A IfxInt32 structure with the value of **source**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt32.Parse Method:**

Converts the supplied String to IfxInt32.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Function Parse (szInt32 As String) As IfxInt32
[C#]
public static IfxInt32 Parse (string szInt32)
[C++]
public:
```

```
static IfxInt32 Parse (string szInt32)
[JScript]
public static function Parse (szInt32 String ) : IfxInt32
```

**Parameters**

**szInt32**
A String value to be converted to IfxInt32.

**Return value**

A IfxInt32 structure with the numeric value of **szInt32**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt32.ToString Method:**

Returns a string that represents the IfxInt32 structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxInt32 structure.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt32 Properties

The properties of the IfxInt32 structure are listed here.

## Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxInt32 structure is null. |
| Value | Gets the value stored in the IfxInt32 structure. |

**Reference**

"IfxInt32 Members" on page 5-438
"IfxInt32 Structure" on page 5-437
"IBM.Data.Informix Namespace" on page 5-1

**IfxInt32.IsNull Property:**

Gets a value that indicates if the value stored in the IfxInt32 structure is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt32 Structure" on page 5-437
"IBM.Data.Informix Namespace" on page 5-1

**IfxInt32.Value Property:**

Gets the value stored in the IfxInt32 structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public ReadOnly Property Value As Short
[C#]
public short Value {get;}
[C++]
public: __property short get_Value();
[JScript]
public function get Value() : short;
```

### Property value

A short integer representing the IfxInt32 instance.

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt32 Structure" on page 5-437

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt64 Structure

Represents the BigInt Informix data type. Encapsulates the long .NET data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Structure IfxInt64
[C#]
public struct IfxInt64
[C++]
public value class IfxInt64
```

### Example

[C#] The following example demonstrates how to retrieve a single BIGINT column value from a table.

```
[C#]
  public static string getParam(IfxConnection conn)
  {
    string mySelectQuery = "SELECT * FROM CUSTOMER";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);
    IfxDataReader reader = myCommand.ExecuteReader();

    if (reader.Read())
    {
      IfxInt64 selectValue = reader.GetIfxInt64(0);

      if (!selectValue.IsNull) { return selectValue.ToString(); }
    }

    return "NULL";
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxInt64 Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt64 Members

Represents the BigInt Informix data type. Encapsulates the long .NET data type. The following tables list the members exposed by the IfxInt64 class.

### Public Fields

| Field | Description |
|---|---|
| MaxValue | The maximum value for IfxInt64: 2^63-1. |
| MinValue | The minumum value for IfxInt64: -2^63. |
| Null | Null value for IfxInt64. |
| Zero | Zero value for IfxInt64. |

### Public Constructors

| Constructor | Description |
|---|---|
| IfxInt64 | Initializes a new IfxInt64 structure with the specified value. |

### Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxInt64 structure is null. |
| Value | Gets the value stored in the IfxInt64 structure. |

### Public Methods

| Method | Description |
|---|---|
| Parse | Converts the supplied String to IfxInt64. |

| Method | Description |
|---|---|
| ToString | Returns a string that represents the IfxInt64 structure. |
| op_explicit | Converts the supplied IfxInt64 structure to a long integer. |
| op_implicit | Converts the supplied long integer to IfxInt64. |

**Reference**

"IfxInt64 Structure" on page 5-446
"IBM.Data.Informix Namespace" on page 5-1

## IfxInt64 Fields

The fields of the IfxInt64 class are listed here.

### Public Fields

| Field | Description |
|---|---|
| MaxValue | The maximum value for IfxInt64: 2^63-1. |
| MinValue | The minumum value for IfxInt64: -2^63. |
| Null | Null value for IfxInt64. |
| Zero | Zero value for IfxInt64. |

**Reference**

"IfxInt64 Members" on page 5-447
"IfxInt64 Structure" on page 5-446
"IBM.Data.Informix Namespace" on page 5-1

**IfxInt64.MaxValue Field:**

The maximum value for IfxInt64: 2^63-1.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MaxValue As IfxInt64
[C#]
public static readonly IfxInt64 MaxValue
[C++]
public:
static initonly IfxInt64 MaxValue
[JScript]
public static final var MaxValue () : IfxInt64
```

**Remarks**

The value of this constant is 2^63-1.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt64.MinValue Field:**

The minumum value for IfxInt64: -2^63.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MinValue As IfxInt64
[C#]
public static readonly IfxInt64 MinValue
[C++]
public:
static initonly IfxInt64 MinValue
[JScript]
public static final var MinValue () : IfxInt64
```

**Remarks**

The value of this constant is -2^63.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt64.Null Field:**

Null value for IfxInt64.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxInt64
[C#]
```

```
public static readonly IfxInt64 Null
[C++]
public:
static initonly IfxInt64 Null
[JScript]
public static final var Null () : IfxInt64
```

**Remarks**

The value of this constant is `NULL`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt64.Zero Field:**

Zero value for IfxInt64.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Zero As IfxInt64
[C#]
public static readonly IfxInt64 Zero
[C++]
public:
static initonly IfxInt64 Zero
[JScript]
public static final var Zero () : IfxInt64
```

**Remarks**

The value of this constant is `0`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt64 Constructor

Initializes a new IfxInt64 structure with the specified value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New(value as Long)
[C#]
public IfxInt64(long value);
[C++]
public: IfxInt64(long value);
[JScript]
public function IfxInt64(value : long);
```

### Parameters

**value** A long integer value to populate the IfxInt64 instance.

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt64 Methods

The methods of the IfxInt64 class are listed here.

### Public Methods

| Method | Description |
|---|---|
|  Parse | Converts the supplied String to IfxInt64. |
|  ToString | Returns a string that represents the IfxInt64 structure. |
|  op_explicit | Converts the supplied IfxInt64 structure to a long integer. |
|  op_implicit | Converts the supplied long integer to IfxInt64. |

**Reference**

"IfxInt64 Members" on page 5-447

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt64.op_explicit Method:**

Converts the supplied IfxInt64 structure to a long integer.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Narrowing Operator CType (source As IfxInt64) As Long
[C#]
```

```
public static implicit operator long (IfxInt64 source)
[C++]
public:
static implicit operator long (IfxInt64 source)
```

**Parameters**

**source** A IfxInt64 structure to be converted to a long integer.

**Return value**

A long integer value converted from a IfxInt64 instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt64.op_implicit Method:**

Converts the supplied long integer to IfxInt64.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Widening Operator CType (source As Long) As IfxInt64
[C#]
public static implicit operator IfxInt64 (long source)
[C++]
public:
static implicit operator IfxInt64 (long source)
```

**Parameters**

**source** A long integer value to be converted to IfxInt64.

**Return value**

A IfxInt64 structure with the value of **source**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt64.Parse Method:**

Converts the supplied String to IfxInt64.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Function Parse (szInt64 As String) As IfxInt64
[C#]
public static IfxInt64 Parse (string szInt64)
[C++]
public:
static IfxInt64 Parse (string szInt64)
[JScript]
public static function Parse (szInt64 String ) : IfxInt64
```

**Parameters**

**szInt64**
A String value to be converted to IfxInt64.

**Return value**

A IfxInt64 structure with the numeric value of **szInt64**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt64.ToString Method:**

Returns a string that represents the IfxInt64 structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxInt64 structure.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

## IfxInt64 Properties

The properties of the IfxInt64 class are listed here.

### Public Properties

| Property | Description |
|---|---|
|  IsNull | Gets a value that indicates if the value stored in the IfxInt64 structure is null. |
|  Value | Gets the value stored in the IfxInt64 structure. |

**Reference**

"IfxInt64 Members" on page 5-447

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt64.IsNull Property:**

Gets a value that indicates if the value stored in the IfxInt64 structure is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

**IfxInt64.Value Property:**

Gets the value stored in the IfxInt64 structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Value As Long
[C#]
public long Value {get;}
[C++]
public: __property long get_Value();
[JScript]
public function get Value() : long;
```

**Property value**

A long integer representing the IfxInt64 instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt64 Structure" on page 5-446

"IBM.Data.Informix Namespace" on page 5-1

# IfxNullValueException Class

The exception that is generated when the Value method for the applicable IBM.Data.Informixclasses object is called against a null value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
  System.Exception
    System.SystemException
      IBM.Data.Informix.Exception
        IBM.Data.Informix.IfxNullValueException

## Syntax

```
[Visual Basic]
Public Class IfxNullValueException
[C#]
public class IfxNullValueException
[C++]
public __gc class IfxNullValueException
[JScript]
public class IfxNullValueException
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxNullValueException Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxNullValueException Members

The exception that is generated when the Value method for the applicable IBM.Data.Informix structure or class is called against a null value. The following tables list the members exposed by the IfxNullValueException class.

### Public Properties

| Property | Description |
|---|---|
| HelpLink (inherited from Exception) | Gets or sets a link to the help file associated with this exception. |
| InnerException (inherited from Exception) | Gets the Exception instance that caused the current exception. |
| Message (inherited from Exception | Gets the text describing the error. |
| Source (inherited from Exception | Gets the name of the application or the object that generated the error. |
| StackTrace (inherited from Exception) | Gets a string representation of the frames on the call stack at the time the current exception was thrown. |
| TargetSite (inherited from Exception) | Gets the method that throws the current exception. |

### Public Methods

| Method | Description |
|---|---|
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetBaseException (inherited from Exception) | When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetObjectData (inherited from Object) | Overridden. See Exception.GetObjectData. |

| Method | Description |
|---|---|
| GetType (inherited from Object) | Gets the Type of the current instance. |
| ToString (inherited from Exception) | Overridden. Creates and returns a string representation of the current exception. |

### Protected Properties

| Property | Description |
|---|---|
| HResult (inherited from Exception) | Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. |

### Protected Methods

| Method | Description |
|---|---|
| Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxNullValueException Class" on page 5-455

## IfxParameter Class

Represents a parameter to an IfxCommand and, optionally, its mapping to a DataColumn.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### .NET Framework 2.0, 3.0, 3.5, and 4.0 inheritance hierarchy

System.Object
  System.MarshalByRefObject
    System.Data.Common.DbParameter
      IBM.Data.Informix.IfxParameter

### .NET Framework 2.0, 3.0, 3.5, and 4.0 syntax

```
[Visual Basic]
NotInheritable Public Class IfxParameter
   Inherits DbParameter
   Implements IDbDataParameter, IDataParameter, ICloneable
[C#]
public sealed class IfxParameter : DbParameter,
```

```
   IDbDataParameter, IDataParameter, ICloneable
[C++]
public __gc __sealed class IfxParameter : public
   DbParameter, IDbDataParameter, IDataParameter, ICloneable
[JScript]
public class IfxParameter extends DbParameter implements
   IDbDataParameter, IDataParameter, ICloneable
```

## Remarks

You can assign two types of values to parameters:

- Values that you assign to a DB2Type enumerated type
- Values in IBM.Data.IfxTypes namespace classes and structures

The parameter names are not case sensitive.

## Example

[Visual Basic, C#] The following examples create multiple instances of the IfxParameter class by using the IfxParameterCollection within the IfxDataAdapter. These parameters are used to select data from the database and place the data in the DataSet. It is assumed that a DataSet and a IfxDataAdapter have already been created with the appropriate schema, commands, and connection.

```
[Visual Basic]
Public Sub AddIfxParameters()
    ' ...
    ' create myDataSet and myDataAdapter
    ' ...
myDataAdapter.SelectCommand.Parameters.Add(
  "CategoryName", IfxType.VarChar, 80).Value = "toasters"
myDataAdapter.SelectCommand.Parameters.Add(
  "SerialNum", IfxType.Integer).Value = 239
myDataAdapter.Fill(myDataSet)
End Sub 'AddIfxParameters

[C#]
public void AddIfxParameters()
 {
 // ...
 // create myDataSet and myDataAdapter
 // ...
   myDataAdapter.SelectCommand.Parameters.Add(
     "CategoryName", IfxType.VarChar, 80).Value = "toasters";
   myDataAdapter.SelectCommand.Parameters.Add(
     "SerialNum", IfxType.Integer).Value = 239;
   myDataAdapter.Fill(myDataSet);
 }
```

The addEmp procedure is then called by using a named argument along with its value in a CALL statement:

```
CALL addEmp (empNo => 5, empName => 'John', empDeptNo =>
2, empAddr => 'San Jose, CA')
```

## Query limitation

You cannot use a positioned parameter marker (?) and a named parameter marker (for example, : paramEmpAddr) in the same query.

## Thread safety

Public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxParameter Members"

"IfxType Enumeration" on page 5-608

"IBM.Data.Informix Namespace" on page 5-1

"IBM.Data.Informix Namespace" on page 5-1

## IfxParameter Members

IfxParameter overview class members

### Public Fields

| Name | Description |
|---|---|
|  Default | IfxParameter.Default Field gets an IfxParameter object with the IsDefault property set to `true`. |
|  Unassigned | IfxParameter.Unassigned Field gets an IfxParameter object with the IsUnassigned property set to `true`. |

### Public Constructors

| Name | Description |
|---|---|
| IfxParameter() | Initializes a new instance of the IfxParameter class. |
| IfxParameter(String, Object) | Initializes a new instance of the IfxParameter class with the parameter name and an object holding a value to be passed as a parameter (for example, another IfxParameter instance, or instances of IBM.Data.IfxTypes structures or classes). |
| IfxParameter(String, IfxType) | Initializes a new instance of the IfxParameter class with the parameter name and data type. |
| IfxParameter(String, IfxType, Integer) | Initializes a new instance of the IfxParameter class with the parameter name, data type, and width. |
| IfxParameter(String, IfxType, Integer, String) | Initializes a new instance of the IfxParameter class with the parameter name, data type, width, and source column name. |
| IfxParameter(String, IfxType, Integer, ParameterDirection, Boolean, Byte, Byte, String, DataRowVersion, Object) | Initializes a new instance of the IfxParameter class with the parameter name, data type, width, source column name, parameter direction, numeric precision, and other properties. |

## Public Properties

| Property | Description |
|---|---|
| IfxType | Gets or sets the IfxType of the parameter. |
| IfxTypeOutput | Determines if the output-only parameter values associated with the IfxParameter is returned as native DB2 data types (specifically, classes and structures in the IBM.Data.IfxTypes namespace). |
| DbType | Gets or sets the DbType of the parameter. |
| Direction | Gets or sets a value indicating whether the parameter is input-only, output-only, bidirectional, or a stored procedure return value parameter. |
| IsNullable | Gets or sets a value indicating whether the parameter accepts `null` values. |
| ParameterName | Gets or sets the name of the IfxParameter. |
| Precision | Gets or sets the maximum number of digits used to represent the Value property. |
| Scale | Gets or sets the number of decimal places to which Value is resolved. |
| Size | Gets or sets the maximum size, in bytes, of the data within the column. |
| SourceColumn | Gets or sets the name of the source column mapped to the DataSet and used for loading or returning the Value . |
| SourceColumnNullMapping | Gets or sets a value indicating whether the column is nullable. |
| SourceVersion | Gets or sets the DataRowVersion to use when loading Value . |
| Value | Gets or sets the value of the parameter. |

## Public Methods

| Method | Description |
|---|---|
| CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |

| Method | Description |
|---|---|
| GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| ResetDbType | Overridden. Resets the data type associated with this IfxParameter instance. |
| ToString | Overridden. Gets a string containing the ParameterName . |

### Protected Methods

| Method | Description |
|---|---|
| Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxParameter Class" on page 5-457
"IBM.Data.Informix Namespace" on page 5-1

## IfxParameter Fields

The fields of the IfxParameter class are listed here.

### Public Fields

| Name | Description |
|---|---|
| Default | IfxParameter.Default Field gets an IfxParameter object with the IsDefault property set to `true`. |
| Unassigned | IfxParameter.Unassigned Field gets an IfxParameter object with the IsUnassigned property set to `true`. |

**Reference**

"IfxParameter Class" on page 5-457
"IfxParameter Members" on page 5-459
"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter.Default Field:**

IfxParameter.Default Field gets an IfxParameter object with the IsDefault property set to `true`.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Default As IfxParameter
[C#]
public static readonly IfxParameter Default
[C++]
public:
static initonly IfxParameter^ Default
[JScript]
public static final var Default () : IfxParameter
```

**Remarks**

This static field can be used to create a parameter set up to use the data server defined default value.

**Example**

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxParameter Class" on page 5-457
"IfxParameter Members" on page 5-459
"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter.Unassigned Field:**

IfxParameter.Unassigned Field gets an IfxParameter object with the IsUnassigned property set to `true`.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Unassigned As IfxParameter
[C#]
public static readonly IfxParameter Unassigned
[C++]
public:
static initonly IfxParameter^ Unassigned
[JScript]
public static final var Unassigned () : IfxParameter
```

**Remarks**

This static field can be used to create a parameter set up to use the data server defined default value.

**Example**

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

## IfxParameter Constructor

Initializes a new instance of the IfxParameter class.

### Public Constructors

| Name | Description |
|---|---|
| IfxParameter() | Initializes a new instance of the IfxParameter class. |
| IfxParameter(String, Object) | Initializes a new instance of the IfxParameter class with the parameter name and an object holding a value to be passed as a parameter (for example, another IfxParameter instance, or instances of IBM.Data.IfxTypes structures or classes). |
| IfxParameter(String, IfxType) | Initializes a new instance of the IfxParameter class with the parameter name and data type. |
| IfxParameter(String, IfxType, Integer) | Initializes a new instance of the IfxParameter class with the parameter name, data type, and width. |
| IfxParameter(String, IfxType, Integer, String) | Initializes a new instance of the IfxParameter class with the parameter name, data type, width, and source column name. |
| IfxParameter(String, IfxType, Integer, ParameterDirection, Boolean, Byte, Byte, String, DataRowVersion, Object) | Initializes a new instance of the IfxParameter class with the parameter name, data type, width, source column name, parameter direction, numeric precision, and other properties. |

### Example

[Visual Basic, C#] The following example creates a IfxParameter and displays the ParameterName.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of the IfxParameter constructor. For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter("Description", IfxType.VarChar, _
 11, ParameterDirection.Output, True, 0, 0, "Description", DataRowVersion.Current, _
 "garden hose")
    MessageBox.Show(myParameter.ToString())
End Sub 'CreateIfxParameter

[C#]
public void CreateIfxParameter()
 {
    IfxParameter myParameter = new IfxParameter("Description",IfxType.VarChar,
                11,ParameterDirection.Output,true,0,0,"Description",
                DataRowVersion.Current,"garden hose");
    MessageBox.Show(myParameter.ToString());
 }
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter Constructor ():**

Initializes a new instance of the IfxParameter class.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New()
[C#]
public IfxParameter();
[C++]
public: IfxParameter();
[JScript]
public function IfxParameter();
```

**Remarks**

The base constructor initializes all fields to their default values.

**Example**

[Visual Basic, C#] The following example creates a IfxParameter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter()
    myParameter.ParameterName = "Description"
    myParameter.IfxType = IfxType.VarChar
    myParameter.Direction = ParameterDirection.Output
    myParameter.Size = 88
End Sub 'CreateIfxParameter

[C#]
 public void CreateIfxParameter()
 {
```

```
    IfxParameter myParameter = new IfxParameter();
    myParameter.ParameterName = "Description";
    myParameter.IfxType = IfxType.VarChar;
    myParameter.Direction = ParameterDirection.Output;
    myParameter.Size = 88;
}
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

"IfxParameter Constructor" on page 5-463

**IfxParameter Constructor (String, Object):**

Initializes a new instance of the IfxParameter class with the parameter name and an object holding a value to be passed as a parameter (for example, another IfxParameter instance, or instances of IBM.Data.IfxTypes structures or classes).

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New( _
   ByVal name As String, _
   ByVal value As Object _
)
[C#]
public IfxParameter(
   string name,
   object value
);
[C++]
public: IfxParameter(
   String* name,
   Object* value
);
[JScript]
public function IfxParameter(
   name : String,
   value : Object
);
```

**Parameters**

**name** The name of the parameter.

**value** An object holding a value to be passed as a parameter (for example, another IfxParameter instance, or instances of IBM.Data.IfxTypes structures or classes).

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Constructor" on page 5-463

"IfxParameter Members" on page 5-459

"IfxType Enumeration" on page 5-608

"IBM.Data.Informix Namespace" on page 5-1

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter Constructor (String, IfxType):**

Initializes a new instance of the IfxParameter class with the parameter name and data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New( _
   ByVal name As String, _
   ByVal type As IfxType
_
)
[C#]
public IfxParameter(
   string name,
   IfxType
 type
);
[C++]
public: IfxParameter(
   String* name,
   IfxType
 type
);
[JScript]
public function IfxParameter(
   name : String,
   type : IfxType

);
```

**Parameters**

**name** The name of the parameter.

**type** One of the IfxType enumeration values.

**Example**

[Visual Basic, C#] The following example creates a IfxParameter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter("Description", IfxType.VarChar)
    myParameter.Direction = ParameterDirection.Output
    myParameter.Size = 88
End Sub 'CreateIfxParameter

[C#]
 public void CreateIfxParameter()
 {
    IfxParameter myParameter = new IfxParameter("Description",IfxType.VarChar);
    myParameter.Direction = ParameterDirection.Output;
    myParameter.Size = 88;
 }
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IfxParameter Constructor" on page 5-463

"IfxType Enumeration" on page 5-608

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter Constructor (String, IfxType, Int32):**

Initializes a new instance of the IfxParameter class with the parameter name, data type, and width.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New( _
   ByVal name As String, _
   ByVal type As IfxType
, _
   ByVal size As Integer _
)
[C#]
public IfxParameter(
   string name,
   IfxType
 type,
   int size
);
[C++]
public: IfxParameter(
   String* name,
   IfxType
 type,
   int size
);
[JScript]
public function IfxParameter(
   name : String,
   type : IfxType
,
   size : int
);
```

**Parameters**

**name** The name of the parameter.

**type** One of the IfxType enumeration values.

**size** The width of the parameter.

**Example**

[Visual Basic, C#] The following example creates a IfxParameter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter("Description", IfxType.VarChar, 88)
    myParameter.Direction = ParameterDirection.Output
End Sub 'CreateIfxParameter

[C#]
 public void CreateIfxParameter()
 {
    IfxParameter myParameter = new IfxParameter("Description",IfxType.VarChar,88);
    myParameter.Direction = ParameterDirection.Output;
 }
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IfxParameter Constructor" on page 5-463

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter Constructor (String, IfxType, Int32, String):**

Initializes a new instance of the IfxParameter class with the parameter name, data type, width, and source column name.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New( _
   ByVal name As String, _
   ByVal type As IfxType
, _
   ByVal size As Integer, _
   ByVal sourcecolumn As String _
)
[C#]
public IfxParameter(
   string name,
   IfxType
 type,
   int size,
   string sourcecolumn
);
[C++]
public: IfxParameter(
   String* name,
   IfxType
 type,
   int size,
   String* sourcecolumn
);
[JScript]
public function IfxParameter(
   name : String,
   type : IfxType
,
   size : int,
   sourcecolumn : String
);
```

**Parameters**

**name** The name of the parameter.

**type** One of the IfxType enumeration values.

**size** The width of the parameter.

**sourcecolumn**
The name of the source column.

**Example**

[Visual Basic, C#] The following example creates a IfxParameter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter(
      "Description", IfxType.VarChar, 88, "Description")
      myParameter.Direction = ParameterDirection.Output
End Sub 'CreateIfxParameter

[C#]
 public void CreateIfxParameter()
 {
    IfxParameter myParameter = new IfxParameter("Description",IfxType.VarChar,
                88,"Description");
    myParameter.Direction = ParameterDirection.Output;
 }
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IfxParameter Constructor" on page 5-463

"IfxType Enumeration" on page 5-608

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter Constructor (String, IfxType, Int32, ParameterDirection, Boolean, Byte, Byte, String, DataRowVersion, Object):**

Initializes a new instance of the IfxParameter class with the parameter name, data type, width, source column name, parameter direction, numeric precision, and other properties.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New( _
   ByVal parameterName As String, _
   ByVal IfxType As IfxType
, _
   ByVal size As Integer, _
   ByVal parameterDirection As ParameterDirection, _
   ByVal isNullable As Boolean, _
   ByVal precision As Byte, _
   ByVal scale As Byte, _
   ByVal srcColumn As String, _
```

```
   ByVal srcVersion As DataRowVersion, _
   ByVal value As Object _
)
[C#]
public IfxParameter(
   string parameterName,
   IfxType
 IfxType,
   int size,
   ParameterDirection parameterDirection,
   bool isNullable,
   byte precision,
   byte scale,
   string srcColumn,
   DataRowVersion srcVersion,
   object value
);
[C++]
public: IfxParameter(
   String* parameterName,
   IfxType
 IfxType,
   int size,
   ParameterDirection parameterDirection,
   bool isNullable,
   unsigned char precision,
   unsigned char scale,
   String* srcColumn,
   DataRowVersion srcVersion,
   Object* value
);
[JScript]
public function IfxParameter(
   parameterName : String,
   IfxType : IfxType
,
   size : int,
   parameterDirection : ParameterDirection,
   isNullable : Boolean,
   precision : Byte,
   scale : Byte,
   srcColumn : String,
   srcVersion : DataRowVersion,
   value : Object
);
```

**Parameters**

**parameterName**
The name of the parameter.

**IfxType**
One of the IfxType values.

**size** The width of the parameter.

**parameterDirection**
One of the System.Data.ParameterDirection values.

**isNullable**
`true` if the value of the field can be null; otherwise, `false`.

**precision**
The total number of digits to the left and right of the decimal point to which IfxParameterClassValue is resolved.

**scale** The total number of decimal places to which IfxParameterClassValue is resolved.

**srcColumn**
The name of the source column.

**srcVersion**
One of the System.Data.DataRowVersion values.

**value** An object holding a value to be passed as a parameter (for example, another IfxParameter instance, or instances of IBM.Data.IfxTypes structures or classes).

**Example**

[Visual Basic, C#] The following example creates a IfxParameter and displays the ParameterName.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter("Description", IfxType.VarChar, 11,
      ParameterDirection.Output, True, 0, 0, "Description",
          DataRowVersion.Current, "garden hose")
    MessageBox.Show(myParameter.ToString())
End Sub 'CreateIfxParameter

[C#]
public void CreateIfxParameter()
 {
    IfxParameter myParameter = new IfxParameter("Description",IfxType.VarChar,
                11,ParameterDirection.Output,true,0,0,"Description",
                DataRowVersion.Current,"garden hose");
    MessageBox.Show(myParameter.ToString());
 }
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IfxParameter Constructor" on page 5-463

"IfxType Enumeration" on page 5-608

"IBM.Data.Informix Namespace" on page 5-1

"IBM.Data.Informix Namespace" on page 5-1

## IfxParameter Methods

The methods of the IfxParameter class are listed here. For a complete list of IfxParameter class members, see the IfxParameter Members topic.

### Public Methods

| Method | Description |
|---|---|
| CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |

| Method | Description |
|---|---|
| GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| ResetDbType | Overridden. Resets the data type associated with this IfxParameter instance. |
| ToString | Overridden. Gets a string containing the ParameterName . |

## Protected Methods

| Method | Description |
|---|---|
| Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxParameter Class" on page 5-457

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter.ICloneable.Clone Method:**

This member supports the Microsoft(R) .NET Framework infrastructure and is not intended to be used directly from your code.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Function Clone() As Object Implements ICloneable.Clone
[C#]
object ICloneable.Clone();
[C++]
Object* ICloneable::Clone();
[JScript]
function ICloneable.Clone() : Object;
```

**Reference**

"IfxParameter Class" on page 5-457
"IfxParameter Members" on page 5-459
"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter.ResetDbType Method:**

Resets the data type associated with this IfxParameter instance.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Sub ResetDbType
[C#]
public override void ResetDbType ()
[C++]
public:
virtual void ResetDbType () override
[JScript]
public override function ResetDbType ()
```

**Remarks**

When executing a IfxCommand, you can assign parameter types explicitly, or implicitly allow the parameter type to be determined based on the the value passed to the parameter. If an explicit data type assignment is made, a call to this method will reset the parameter type for a IfxParameter so that it's based on the value passed to the parameter.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxParameter Class" on page 5-457
"IfxParameter Members" on page 5-459
"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter.ToString Method:**

Gets a string containing the ParameterName.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overrides Public Function ToString() As String
[C#]
public override string ToString();
[C++]
```

```
public: String* ToString();
[JScript]
public override function ToString() : String;
```

**Return value**

A string containing the ParameterName.

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

## IfxParameter Properties

The properties of the IfxParameter class are listed here. For a complete list of IfxParameter class members, see the IfxParameter Members topic.

### Public Properties

| Property | Description |
|---|---|
| IfxType | Gets or sets the IfxType of the parameter. |
| IfxTypeOutput | Determines if the output-only parameter values associated with the IfxParameter is returned as native DB2 data types (specifically, classes and structures in the IBM.Data.IfxTypes namespace). |
| DbType | Gets or sets the DbType of the parameter. |
| Direction | Gets or sets a value indicating whether the parameter is input-only, output-only, bidirectional, or a stored procedure return value parameter. |
| IsNullable | Gets or sets a value indicating whether the parameter accepts `null` values. |
| ParameterName | Gets or sets the name of the IfxParameter. |
| Precision | Gets or sets the maximum number of digits used to represent the Value property. |
| Scale | Gets or sets the number of decimal places to which Value is resolved. |
| Size | Gets or sets the maximum size, in bytes, of the data within the column. |
| SourceColumn | Gets or sets the name of the source column mapped to the DataSet and used for loading or returning the Value . |
| SourceColumnNullMapping | Gets or sets a value indicating whether the column is nullable. |
| SourceVersion | Gets or sets the DataRowVersion to use when loading Value . |
| Value | Gets or sets the value of the parameter. |

**Reference**

"IfxParameter Class" on page 5-457

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter.IfxTypeOutput Property:**

Determines if the output-only parameter values associated with the IfxParameter will be returned as native DB2 data types (specifically, classes and structures in the IBM.Data.IfxTypes namespace).

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
Public Overrides Property IfxTypeOutput As Boolean
[C#]
public override bool IfxTypeOutput { get; set; }
[C++]
public:
virtual property bool IfxTypeOutput {
   bool get () override;
   void set (bool value) override;
}
```

**Property value**

If the parameter is set to output-only, this boolean value determines if column values will be returned as native DB2 data type instances (for example, IfxString).

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter.IfxType Property:**

Gets or sets the IfxType of the parameter.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property IfxType As IfxType

[C#]
public IfxType
 IfxType {get; set;}
```

```
[C++]
public: __property IfxType
 get_IfxType();
public: __property void set_IfxType(IfxType
);
[JScript]
public function get IfxType() : IfxType
;
public function set IfxType(IfxType
);
```

**Property value**

An IfxType value that is the IfxType of the parameter. The default is VarChar.

**Remarks**

The IfxType and DbType properties are linked. Therefore, setting the DbType changes the IfxType to a supporting IfxType. Conversely, setting the IfxType changes the DbType to a supporting DbType.

For a list of the supported data types, see the appropriate IfxType member.

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IfxType Enumeration" on page 5-608

"IBM.Data.Informix Namespace" on page 5-1

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter.DbType Property:**

Gets or sets the DbType of the parameter.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property DbType As DbType  Implements IDataParameter.DbType
[C#]
public DbType DbType {get; set;}
[C++]
public: __property DbType get_DbType();
public: __property void set_DbType(DbType);
[JScript]
public function get DbType() : DbType;
public function set DbType(DbType);
```

**Implements:**
IDataParameter.DbType

**Property value**

One of the System.Data.DbType values. The default is String.

**Exceptions**

| Exception type | Condition |
|---|---|
| ArgumentOutOfRangeException | The property was not set to a valid DbType. |

**Remarks**

The IfxType and DbType properties are linked. Therefore, setting the IfxType changes the DbType to a supporting DbType. Conversely, setting the DbType changes the IfxType to a supporting IfxType.

For a list of the supported data types, see the appropriate DbType member.

**Note:** Object is not supported by the IBM Data Server Provider for .NET.

**Example**

[Visual Basic, C#] The following example creates a IfxParameter and sets some of its properties, including the DbType Property.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter()
    myParameter.ParameterName = "Description"
    myParameter.DbType = DbType.String
    myParameter.Direction = ParameterDirection.Output
    myParameter.Size = 88
End Sub 'CreateIfxParameter
[C#]
public void CreateIfxParameter()
 {
    IfxParameter myParameter = new IfxParameter();
    myParameter.ParameterName = "Description";
    myParameter.DbType = DbType.String;
    myParameter.Direction = ParameterDirection.Output;
    myParameter.Size = 88;
 }
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IfxType Enumeration" on page 5-608

"IBM.Data.Informix Namespace" on page 5-1

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter.Direction Property:**

Gets or sets a value indicating whether the parameter is input-only, output-only, bidirectional, or a stored procedure return value parameter.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Property Direction As ParameterDirection  Implements _
   IDataParameter.Direction
[C#]
public ParameterDirection Direction {get; set;}
[C++]
public: __property ParameterDirection get_Direction();
public: __property void set_Direction(ParameterDirection);
[JScript]
public function get Direction() : ParameterDirection;
public function set Direction(ParameterDirection);
```

**Implements:**

IDataParameter.Direction

### Property value

One of the System.Data.ParameterDirection values. The default is Input.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentException | The property was not set to one of the valid ParameterDirection values. |

### Remarks

If the ParameterDirection is Output, and execution of the associated IfxCommand does not return a value, the IfxParameter will contain a null value. Null values are handled using the DBNull class. The Output, InputOut, and ReturnValue parameters are updated after the IfxCommand has been executed.

### Example

[Visual Basic, C#] The following example creates a IfxParameter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter("Description", IfxType.VarChar, 88)
    myParameter.Direction = ParameterDirection.Output
End Sub 'CreateIfxParameter

[C#]
public void CreateIfxParameter()
 {
    IfxParameter myParameter = new IfxParameter("Description", IfxType.VarChar, 88);
    myParameter.Direction = ParameterDirection.Output;
 }
```

### Reference

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

## IfxParameter.IsNullable Property:

Gets or sets a value indicating whether the parameter accepts null values.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property IsNullable As Boolean  Implements _
   IDataParameter.IsNullable
[C#]
public bool IsNullable {get; set;}
[C++]
public: __property bool get_IsNullable();
public: __property void set_IsNullable(bool);
[JScript]
public function get IsNullable() : Boolean;
public function set IsNullable(Boolean);
```

**Implements:**
IDataParameter.IsNullable

**Property value**

`true` if null values are accepted; otherwise, `false`. The default is `false`.

**Remarks**

Null values are handled using the System.DBNull class.

**Example**

[Visual Basic, C#] The following example creates a IfxParameter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter("Description", IfxType.VarChar, 88)
    myParameter.IsNullable = True
    myParameter.Direction = ParameterDirection.Output
End Sub 'CreateIfxParameter

[C#]
public void CreateIfxParameter()
 {
    IfxParameter myParameter = new IfxParameter("Description", IfxType.VarChar, 88);
    myParameter.IsNullable = true;
    myParameter.Direction = ParameterDirection.Output;
 }
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter.ParameterName Property:**

Gets or sets the name of the IfxParameter.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property ParameterName As String  Implements _
   IDataParameter.ParameterName
[C#]
public string ParameterName {get; set;}
[C++]
public: __property String* get_ParameterName();
public: __property void set_ParameterName(String*);
[JScript]
public function get ParameterName() : String;
public function set ParameterName(String);
```

**Implements:**
IDataParameter.ParameterName

**Property value**

The name of the IfxParameter . The default is an empty string ("").

**Remarks**

When using named parameters (prefixed with @) or host variables (prefixed with :) in a IfxParameterCollection, you must set ParameterName before executing a IfxCommand that relies on parameters. If positioned parameters are used, any parameter names will be ignored during parameter object binding.

Parameter marker names are case-insensitive, must be prefixed by the symbol '@'or by a colon ':', and can be made up of any symbol that can be used as part of an SQL identifier. For details regarding SQL identifiers, see the topic: “Identifiers” in the DB2 data server documentation. Using more than one type of parameter marker within the same statement is not supported. This means that a statement using positioned parameters cannot contain named parameters or host variables. Likewise, a statement using named parameters cannot contain host variables or positioned parameters

Support for host variables, prefixed by a colon ':', is disabled by default. To enable host variable support the HostVarParameters property must be set to `TRUE` in the connection string.

**Example**

[Visual Basic, C#] The following example assumes that the database has a table name MyTable and a stored procedure named MyProc, that is defined as:

```
create table MyTable (col1 int, col2 double, col3 decimal)
create proc MyProc (p1 int, p2 double, p3 decimal) language sql LABEL1:
  begin insert into MyTable values (p1, p2, p3); end
```

[Visual Basic, C#] The example creates parameters and calls the MyProc stored procedure.

```
[Visual Basic]
Public Sub CallMyProc()
    ' Create and open IfxConnection myConnection.
    Dim myCommand As IfxCommand = myConnection.CreateCommand()
    myCommand.CommandText = "call MyProc(@param1,@param2,@param3)"
```

```
    Dim p1 As New IfxParameter()
    p1.ParameterName = "@param1"
    p1.IfxType = IfxType.Integer
    p1.Value = 1
    Dim p2 As New IfxParameter()
    p2.ParameterName = "@param2"
    p2.IfxType = IfxType.Double
    p2.Value = 2
    Dim p3 = new IfxParameter()
    p3.ParameterName = "@param3"
    p3.IfxType = IfxType.Decimal
    p3.Value = 3

    // Add parameters to the myCommand.Parameters collection
    // and then execute the command.
End Sub ' CallMyProc

[C#]
public void CallMyProc()
{
    // Create and open IfxConnection myConnection.

    IfxCommand myCommand = myConnection.CreateCommand();
    myCommand.CommandText = "call MyProc(@param1,@param2,@param3)";
    Dim p1 As New IfxParameter();
    p1.ParameterName = "@param1";
    p1.IfxType = IfxType.Integer;
    p1.Value = 1;
    Dim p2 As New IfxParameter();
    p2.ParameterName = "@param2";
    p2.IfxType = IfxType.Double;
    p2.Value = 2;
    Dim p3 = new IfxParameter();
    p3.ParameterName = "@param3";
    p3.IfxType = IfxType.Decimal;
    p3.Value = 3;

    // Add parameters to the myCommand.Parameters collection
    // and then execute the command.
}
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter.Precision Property:**

Gets or sets the maximum number of digits used to represent the Value property.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Precision As Byte  Implements _
   IDbDataParameter.Precision
[C#]
public byte Precision {get; set;}
[C++]
public: __property unsigned char get_Precision();
```

```
public: __property void set_Precision(unsigned char);
[JScript]
public function get Precision() : Byte;
public function set Precision(Byte);
```

**Implements:**

IDbDataParameter.Precision

**Property value**

The maximum number of digits used to represent the Value property. The default value is 0.

**Remarks**

The Precision property is used only for decimal and numeric input parameters.

**Example**

[Visual Basic, C#] The following example creates a IfxParameter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter("Price", IfxType.Decimal)
    myParameter.Value = 3.1416
    myParameter.Precision = 8
    myParameter.Scale = 4
End Sub 'CreateIfxParameter

[C#]
public void CreateIfxParameter()
 {
    IfxParameter myParameter = new IfxParameter("Price", IfxType.Decimal);
    myParameter.Value = 3.1416;
    myParameter.Precision = 8;
    myParameter.Scale = 4;
 }
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

"IfxParameter.Scale Property"

"IfxParameter.Size Property" on page 5-483

**IfxParameter.Scale Property:**

Gets or sets the number of decimal places to which Value is resolved.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Scale As Byte  Implements IDbDataParameter.Scale
[C#]
public byte Scale {get; set;}
```

```
[C++]
public: __property unsigned char get_Scale();
public: __property void set_Scale(unsigned char);
[JScript]
public function get Scale() : Byte;
public function set Scale(Byte);
```

**Implements:**

IDbDataParameter.Scale

**Property value**

The number of decimal places to which Value is resolved. The default is 0.

**Remarks**

The Scale property is used only for decimal and numeric input parameters.

**Example**

[Visual Basic, C#] The following example creates a IfxParameter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter("Price", IfxType.Decimal)
    myParameter.Value = 3.1416
    myParameter.Precision = 8
    myParameter.Scale = 4
End Sub 'CreateIfxParameter

[C#]
public void CreateIfxParameter()
 {
    IfxParameter myParameter = new IfxParameter("Price", IfxType.Decimal);
    myParameter.Value = 3.1416;
    myParameter.Precision = 8;
    myParameter.Scale = 4;
 }
```

**Reference**


"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

"IfxParameter.Precision Property" on page 5-481

"IfxParameter.Size Property"


**IfxParameter.Size Property:**

Gets or sets the maximum size, in bytes, of the data within the column.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Size As Integer  Implements IDbDataParameter.Size
[C#]
public int Size {get; set;}
```

```
[C++]
public: __property int get_Size();
public: __property void set_Size(int);
[JScript]
public function get Size() : int;
public function set Size(int);
```

**Implements:**

IDbDataParameter.Size

**Property value**

The maximum size of the data to transmit to the server.

**Remarks**

The Size property is used for binary and string types.

For variable-length data types, the Size property describes the maximum amount of data to transmit to the server. For example, for a string value, the Size property could be used to limit the amount of data sent to the server to the first one hundred bytes.

For nonstring data types and ANSI string data, the Size property refers to the number of bytes. For Unicode string data, the Size property refers to the number of characters. The count for strings does not include the terminating character.

For output, bidirectional (inout), and return values, you must set the value of Size. This is not required for input parameters. If input parameters are not explicitly set, the value of Size will be inferred from the actual size of the specified parameter value.

For fixed-width data types, the value of Size is ignored.

**Example**

[Visual Basic, C#] The following example creates a IfxParameter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myValue As String = "12 foot scarf - multiple colors, one previous owner"
    Dim myParameter As New IfxParameter("Description", IfxType.VarChar)
    myParameter.Direction = ParameterDirection.Input
    myParameter.Size = myValue.Length
    myParameter.Value = myValue
End Sub 'CreateAdoParameter

[C#]
public void CreateIfxParameter()
 {
    string myValue = "12 foot scarf - multiple colors, one previous owner";
    IfxParameter myParameter = new IfxParameter("Description", IfxType.VarChar);
    myParameter.Direction = ParameterDirection.Input;
    myParameter.Size = myValue.Length;
    myParameter.Value = myValue;
 }
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

### IfxParameter.SourceColumnNullMapping Property:

Gets or sets a value indicating whether or not the column is nullable.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

#### Syntax

```
[Visual Basic]
Public Overrides Property SourceColumnNullMapping As Boolean
[C#]
public override bool SourceColumnNullMapping { get; set; }
[C++]
public:
virtual property bool SourceColumnNullMapping {
 bool get () override;
 void set (bool value) override;
}
[JScript]
public override function get SourceColumnNullMapping () : boolean
public override function set SourceColumnNullMapping (value : boolean)
```

#### Property value

Gets or sets a boolean value indicating whether or not the column is nullable. `true` indicates that the column is nullable. `false` indicates that the column is not nullable.

#### Version information

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

### IfxParameter.SourceColumn Property:

Gets or sets the name of the source column mapped to the DataSet and used for loading or returning the Value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

#### Syntax

```
[Visual Basic]
Public Property SourceColumn As String  Implements _
   IDataParameter.SourceColumn
[C#]
public string SourceColumn {get; set;}
[C++]
```

```
public: __property String* get_SourceColumn();
public: __property void set_SourceColumn(String*);
[JScript]
public function get SourceColumn() : String;
public function set SourceColumn(String);
```

**Implements:**

IDataParameter.SourceColumn

**Property value**

The name of the source column that will be used to set the value of this parameter. The default is an empty string ("").

**Remarks**

When SourceColumn is set to anything other than an empty string, the value of the parameter is retrieved from the column with the SourceColumn name. If Direction is set to Input, the value is taken from the DataSet. If Direction is set to Output, the value is taken from the database. A Direction of InputOutput is a combination of both.

**Example**

[Visual Basic, C#] The following example assumes that the database has a table name MyTable and a stored procedure named MyProc, that is defined as:

```
create table MyTable (col1 int, col2 double, col3 timestamp)
create proc MyProc (p1 int, p2 double, p3 timestamp OUT) language sql LABEL1:
  begin insert into MyTable (col1, col2) values (p1, p2); end
```

[Visual Basic, C#] The example creates parameters and calls the MyProc stored procedure.

```
[Visual Basic]
Public Sub CallMyProc()
    ' Create and open IfxConnection myConnection.
 Dim myCommand As IfxCommand = myConnection.CreateCommand()

    myCommand.CommandText = "{ call MyProc(?,?,?) }"

    Dim p1 As New IfxParameter("Name1", IfxType.Integer)
    p1.Value = 1
    p1.Direction = ParameterDirection.Input
    Dim p2 As New IfxParameter("Name2", IfxType.Double)
    p2.Value = 2
    p2.Direction = ParameterDirection.Input
    Dim p3 As NewIfxParameter("Name3", IfxType.Decimal)
    p3.SourceColumn = "col3"
    p3.Direction = ParameterDirection.Output

    // Add parameters to the myCommand.Parameters collection
    // in the order in which they were created and then execute the command.
End Sub ' CallMyProc

[C#]
public void CallMyProc()
  {
    // Create and open IfxConnection myConnection.
       IfxCommand myCommand = myConnection.CreateCommand();
       myCommand.CommandText = "call MyProc(?,?,?)";

       IfxParameter p1 = new IfxParameter("Name1", IfxType.Integer);
       p1.Value = 1;
```

```
        p1.Direction = ParameterDirection.Input;

        IfxParameter p2 = new IfxParameter("Name2", IfxType.Double);
        p2.Value = 2;
        p2.Direction = ParameterDirection.Input;

        IfxParameter p3 = new IfxParameter("Name3", IfxType.Decimal);
        p3.SourceColumn = "col3";
        p3.Direction = ParameterDirection.Output;
    // Add parameters to the myCommand.Parameters collection
    // in the order in which they were created, and then execute the command.
}
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

### IfxParameter.SourceVersion Property:

Gets or sets the DataRowVersion to use when loading Value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property SourceVersion As DataRowVersion  Implements _
   IDataParameter.SourceVersion
[C#]
public DataRowVersion SourceVersion {get; set;}
[C++]
public: __property DataRowVersion get_SourceVersion();
public: __property void set_SourceVersion(DataRowVersion);
[JScript]
public function get SourceVersion() : DataRowVersion;
public function set SourceVersion(DataRowVersion);
```

**Implements:**
IDataParameter.SourceVersion

**Property value**

One of the System.Data.DataRowVersion values. The default is Current.

**Exceptions**

| Exception type | Condition |
|---|---|
| ArgumentException | The property was not set to one of the DataRowVersion values. |

**Remarks**

The SourceVersion is used by IfxDataAdapter.UpdateCommand during an Update operation to determine whether the parameter value is set to Current or Original. This allows primary keys to be updated. This property is ignored by IfxDataAdapter.InsertCommand and IfxDataAdapter.DeleteCommand.

This property is set to the version of the DataRow used by either the Item property (DataRow indexer), or the GetChildRows method of the DataRow object.

**Example**

[Visual Basic, C#] The following example creates a IfxParameter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter("Description", IfxType.VarChar, 88)
    myParameter.SourceColumn = "Description"
    myParameter.SourceVersion = DataRowVersion.Current
End Sub 'CreateIfxParameter

[C#]
public void CreateIfxParameter()
 {
    IfxParameter myParameter = new IfxParameter("Description", IfxType.VarChar, 88);
    myParameter.SourceColumn = "Description";
    myParameter.SourceVersion = DataRowVersion.Current;
 }
```

**Reference**

"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameter.Value Property:**

Gets or sets the value of the parameter.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Value As Object  Implements IDataParameter.Value
[C#]
public object Value {get; set;}
[C++]
public: __property Object* get_Value();
public: __property void set_Value(Object*);
[JScript]
public function get Value() : Object;
public function set Value(Object);
```

**Implements:**
IDataParameter.Value

**Property value**

An Object that is the value of the parameter. The default value is null.

**Remarks**

For input parameters, the value is bound to the IfxCommand that is sent to the server. For output and return-value parameters, the value is set on completion of the IfxCommand.

When sending a null parameter value to the server, the user can specify DBNull or null.

If the application specifies the database type, the bound value is converted to that type when the provider sends the data to the server. The provider attempts to convert any type of value. Conversion errors may result if the specified type is not compatible with the value.

Both the DbType and IfxType properties can be inferred by setting Value. If applicable, the size, precision and scale will also be inferred from Value.

The Value property is overwritten by the DataAdapter.Update method.

**Example**

[Visual Basic, C#] The following example creates a IfxParameter and sets some of its properties.

```
[Visual Basic]
Public Sub CreateIfxParameter()
    Dim myParameter As New IfxParameter("Description", IfxType.VarChar, 88)
    myParameter.Value = "garden hose"
End Sub 'CreateIfxParameter

[C#]
public void CreateIfxParameter()
 {
    IfxParameter myParameter = new IfxParameter("Description", IfxType.VarChar, 88);
    myParameter.Value = "garden hose";
 }
```

**Reference**


"IfxParameter Class" on page 5-457

"IfxParameter Members" on page 5-459

"IfxParameter.IfxType Property" on page 5-475

"IfxParameter.DbType Property" on page 5-476

"IfxType Enumeration" on page 5-608

"IBM.Data.Informix Namespace" on page 5-1

"IBM.Data.Informix Namespace" on page 5-1


## IfxParameterCollection Class

Represents a collection of parameters relevant to a IfxCommand as well as their respective mappings to columns in a DataSet.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### .NET Framework 2.0 3.0, 3.5, and 4.0 Inheritance hierarchy

System.Object
  System.MarshalByRefObject
    System.Data.Common.DbParameterCollection
      IBM.Data.Informix.IfxParameterCollection

## .NET Framework 2.0 3.0, 3.5, and 4.0 Syntax

```
[Visual Basic]
NotInheritable Public Class IfxParameterCollection
   Inherits DbParameterCollection
   Implements IDataParameterCollection, IList, ICollection, _
   IEnumerable
[C#]
public sealed class IfxParameterCollection : DbParameterCollection,
   IDataParameterCollection, IList, ICollection, IEnumerable
[C++]
public __gc __sealed class IfxParameterCollection : public
   DbParameterCollection, IDataParameterCollection, IList,
   ICollection,
   IEnumerable
[JScript]
public class IfxParameterCollection extends DbParameterCollection
   implements IDataParameterCollection, IList, ICollection,
   IEnumerable
```

## Exceptions

| Exception type | Condition |
|---|---|
| IfxException | Wrong number of parameters. The number of parameters in the collection must be equal to the number of parameter placeholders within the command text. |

## Example

[Visual Basic, C#] The following example assumes the existence of a table named `MyTable` and a stored procedure named `MyProcedure`. These objects are defined with the following statements:

```
create table MyTable (column1 int)
create procedure MyProcedure (p1 int ) language sql LABEL1:
  begin insert into MyTable values (p1); end
```

The example calls the MyProcedure stored procedure.

```
[Visual Basic]
Public Sub AddIfxParameters()
    // Create myConnection and myCommand.

    myCommand.CommandText = "{call MyProcedure (?)}"
    myCommand.Parameters.Add("p1",IfxType.Integer).Value = 100
    myCommand.ExecuteNonQuery()

End Sub 'AddIfxParameters
[C#]
public void AddIfxParameters()
  {
      // Create myConnection and myCommand.

         myCommand.CommandText = "{call MyProcedure (?)}";
         myCommand.Parameters.Add("p1",IfxType.Integer).Value = 100;
         myCommand.ExecuteNonQuery();
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxParameterCollection Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxParameterCollection Members

The following tables list the members exposed by the IfxParameterCollection class.

*Table 5-21. Public Properties*

| Method | Description |
|---|---|
| Count | Gets the number of IfxParameter objects in the collection. |
| IsFixedSize | Gets a value that indicates whether the IfxParameterCollection has a fixed size. |
| IsReadOnly | Gets a value that indicates whether the IfxParameterCollection is read-only. |
| IsSynchronized | Gets a value that indicates whether the IfxParameterCollection is synchronized. |
| this | Overloaded. Gets or sets the IfxParameter with a specified attribute.<br><br>In C#, this property is the indexer for the IfxParameterCollection class. |
| SyncRoot | Gets an object that can be used to synchronize access to the IfxParameterCollection. |

*Table 5-22. Public Methods*

| Method | Description |
|---|---|
| Add | Overloaded. Adds the specified IfxParameter to the collection. |
| AddRange | Overridden. Appends an array of values to the IfxParameterCollection. |
| Clear | Removes all items from the collection. |
| Contains | Overloaded. Gets a value indicating whether a IfxParameter object exists in the collection. |
| CopyTo | Copies IfxParameter objects from the IfxParameterCollection to the specified array. |
| CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetEnumerator | Returns an enumerator that iterates through the IfxParameterCollection. |

*Table 5-22. Public Methods (continued)*

| Method | Description |
|---|---|
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| Insert | Inserts a IfxParameter into the collection at the specified index. |
| Remove | Removes the specified IfxParameter from the collection. |
| RemoveAt | Overloaded. Removes the specified IfxParameter from the collection. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

*Table 5-23. Protected Methods*

| Method | Description |
|---|---|
| Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
| GetParameter | Overloaded. Returns the DbParameter object with the specified name in the IfxParameterCollection. |
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |
| SetParameter | Overloaded. Sets the specified DbParameter to the specified value. |

**Reference**

"IfxParameterCollection Class" on page 5-489

"IBM.Data.Informix Namespace" on page 5-1

## IfxParameterCollection Methods

The methods of the IfxParameterCollection class are listed here. For a complete list of IfxParameterCollection class members, see the IfxParameterCollection Members topic.

*Table 5-24. Public Methods*

| Method | Description |
|---|---|
| Add | Overloaded. Adds the specified IfxParameter to the collection. |
| AddRange | Overridden. Appends an array of values to the IfxParameterCollection. |
| Clear | Removes all items from the collection. |
| Contains | Overloaded. Gets a value indicating whether a IfxParameter object exists in the collection. |
| CopyTo | Copies IfxParameter objects from the IfxParameterCollection to the specified array. |
| CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetEnumerator | Returns an enumerator that iterates through the IfxParameterCollection. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| Insert | Inserts a IfxParameter into the collection at the specified index. |
| Remove | Removes the specified IfxParameter from the collection. |
| RemoveAt | Overloaded. Removes the specified IfxParameter from the collection. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

**Reference**

"IfxParameterCollection Class" on page 5-489
"IBM.Data.Informix Namespace" on page 5-1

**IfxParameterCollection.Add Method:**

Adds the specified IfxParameter to the collection.

**Overload list**

| Name | Description |
|---|---|
| Add(Object) As Integer Implements IList.Add | Adds the specified IfxParameter to the IfxCommand . |
| Add(IfxParameter) As IfxParameter | Adds the specified IfxParameter to the IfxParameterCollection . |
| Add(String, Object) As IfxParameter | Adds a IfxParameter to the IfxCommand given the parameter name and value. |
| Add(String, IfxType) As IfxParameter | Adds a IfxParameter to the IfxCommand given the parameter name and data type. |
| Add(String, IfxType, Integer) As IfxParameter | Adds a IfxParameter to the IfxCommand given the the parameter name, data type, and column width. |
| Add(String, IfxType, Integer, String) As IfxParameter | Adds a IfxParameter to the IfxCommand given the parameter name, data type, column width, and source column name. |

**Example**

[Visual Basic, C#] The following example creates a IfxCommand, a IfxParameterCollection for the command, and adds an instance of IfxParameter to the collection.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of Add. For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub CreateIfxParamColl(myConn As IfxConnection)
    Dim myCommand As IfxCommand =
        New IfxCommand("SELECT * FROM EMPLOYEE WHERE EMPNO = ?", myConn)
    Dim myParamCollection As IfxParameterCollection = myCommand.Parameters
    Dim myNewParameter As IfxParameter =
        myParamCollection.Add("EMPNO", IfxType.VarChar, 6, "EMPNO")
End Sub 'CreateIfxParamColl

[C#]
public void CreateIfxParamColl(IfxConnection myConn) {
    IfxCommand myCommand =
        new IfxCommand("SELECT * FROM EMPLOYEE WHERE EMPNO = ?", myConn);
    IfxParameterCollection myParamCollection = myCommand.Parameters;
    IfxParameter myNewParameter =
        myParamCollection.Add("EMPNO", IfxType.VarChar, 6, "EMPNO");
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

*IfxParameterCollection.Add (Object) Method:*

Adds the specified IfxParameter to the IfxCommand.

**Namespace:**

IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Overloads Public Function Add( _
   ByVal value As Object _
) As Integer Implements IList.Add
[C#]
public int Add(
   object value
);
[C++]
public: __sealed int Add(
   Object* value
);
[JScript]
public function Add(
   value : Object
) : int;
```

**Implements:**
IList.Add

**Parameters**

**value** The IfxParameter to add to the IfxCommand .

**Return value**

An integer value representing the index of the specified IfxParameter object.

**Exceptions**

| Exception type | Condition |
|---|---|
| ArgumentException | The parameter specified by **value** already exists. |
| ArgumentNullException | The **value** parameter is null. |
| InvalidCastException | The **value** parameter is not a IfxParameter . |

**Reference**

"IfxParameterCollection Class" on page 5-489
"IfxParameterCollection Members" on page 5-491
"IBM.Data.Informix Namespace" on page 5-1
"IfxParameterCollection.Add Method" on page 5-493

*IfxParameterCollection.Add (IfxParameter) Method:*

Adds the specified IfxParameter to the IfxCommand.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overloads Public Function Add( _
   ByVal value As IfxParameter
_
) As IfxParameter

[C#]
public IfxParameter
 Add(
   IfxParameter
 value
);
[C++]
public: IfxParameter
* Add(
   IfxParameter
* value
);
[JScript]
public function Add(
   value : IfxParameter

) : IfxParameter
;
```

**Parameters**

**value** The IfxParameter to add to the collection.

**Return value**

The new IfxParameter object.

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

"IfxParameterCollection.Add Method" on page 5-493

*IfxParameterCollection.Add (String, Object) Method:*

Adds a IfxParameter to the IfxCommand given the parameter name and value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overloads Public Function Add( _
   ByVal parameterName As String, _
   ByVal value As Object _
) As IfxParameter

[C#]
public IfxParameter
 Add(
   string parameterName,
   object value
);
```

```
[C++]
public: IfxParameter
* Add(
   String* parameterName,
   Object* value
);
[JScript]
public function Add(
   parameterName : String,
   value : Object
) : IfxParameter
;
```

**Parameters**

**parameterName**
The name of the parameter.

**value** The IfxParameter.Value of the IfxParameter to add to the collection.

**Return value**

The new IfxParameter object.

**Remarks**

When you specify a value, use the **value** parameter, and explicitly set the IfxType as demonstrated in this C# example:

```
IfxCommand rComm = new IfxCommand(null, rConn);
rComm.CommandText = "insert into mytable values (?)";
rComm.Parameters.Add("p1", DBNull.Value);
rComm.Parameters["p1"].IfxType = IfxType.Integer;
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

"IfxParameterCollection.Add Method" on page 5-493

*IfxParameterCollection.Add (String, IfxType) Method:*

Adds a IfxParameter to the IfxCommand given the parameter name and data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overloads Public Function Add( _
   ByVal parameterName As String, _
   ByVal IfxType As IfxType

_
) As IfxParameter

[C#]
public IfxParameter
 Add(
   string parameterName,
   IfxType
```

```
 IfxType
);
[C++]
public: IfxParameter
* Add(
   String* parameterName,
   IfxType
 IfxType
);
[JScript]
public function Add(
   parameterName : String,
   IfxType : IfxType

) : IfxParameter
;
```

**Parameters**

**parameterName**
The name of the parameter.

**IfxType**
One of the IfxType values.

**Return value**

The new IfxParameter object.

**Example**

[Visual Basic, C#] The following example creates a IfxParameterCollection , adds an instance of IfxParameter to the collection, and returns a reference to the new IfxParameter.

```
[Visual Basic]
Public Function CreateIfxParamColl(myConn As IfxConnection) As IfxParameter
    Dim myCommand As IfxCommand = New IfxCommand(
      "SELECT * FROM EMPLOYEE WHERE EMPNO = ?", myConn)
    Dim myParamCollection As IfxParameterCollection = myCommand.Parameters
    Dim myParm As IfxParameter = myParamCollection.Add("EMPNO", IfxType.VarChar)

    Return myParm
End Function 'CreateIfxParamColl

[C#]
public IfxParameter CreateIfxParamColl(IfxConnection myConn) {
    IfxCommand myCommand = new IfxCommand(
      "SELECT * FROM EMPLOYEE WHERE EMPNO = ?", myConn);
    IfxParameterCollection myParamCollection = myCommand.Parameters;
    IfxParameter myParm = myParamCollection.Add("EMPNO", IfxType.VarChar);

    return myParm;
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489
"IfxParameterCollection Members" on page 5-491
"IBM.Data.Informix Namespace" on page 5-1
"IfxParameterCollection.Add Method" on page 5-493

*IfxParameterCollection.Add (String, IfxType, Int32) Method:*

Adds a IfxParameter to the IfxCommand given the the parameter name, data type, and column width.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overloads Public Function Add( _
   ByVal parameterName As String, _
   ByVal IfxType As IfxType
, _
   ByVal size As Integer _
) As IfxParameter

[C#]
public IfxParameter
 Add(
   string parameterName,
   IfxType
 IfxType,
   int size
);
[C++]
public: IfxParameter
* Add(
   String* parameterName,
   IfxType
 IfxType,
   int size
);
[JScript]
public function Add(
   parameterName : String,
   IfxType : IfxType
,
   size : int
) : IfxParameter
;
```

**Parameters**

**parameterName**
The name of the parameter.

**IfxType**
One of the IfxType values.

**size** The width of the column.

**Return value**

The new IfxParameter object.

**Example**

[Visual Basic, C#] The following example creates a IfxParameterCollection , adds an instance of IfxParameter to the collection, and returns a reference to the new IfxParameter.

```
[Visual Basic]
Public Function CreateIfxParamColl(myConn As IfxConnection) As IfxParameter
  Dim myCommand As IfxCommand = New IfxCommand(
    "SELECT * FROM EMPLOYEE WHERE EMPNO = ?", myConn)
  Dim myParamCollection As IfxParameterCollection = myCommand.Parameters
  Dim myNewParameter As IfxParameter = myParamCollection.Add(
    "EMPNO", IfxType.VarChar, 6)

  Return myParm
End Function 'CreateIfxParamColl

[C#]
public IfxParameter CreateIfxParamColl(IfxConnection myConn) {
    IfxCommand myCommand = new IfxCommand(
      "SELECT * FROM EMPLOYEE WHERE EMPNO = ?", myConn);
    IfxParameterCollection myParamCollection = myCommand.Parameters;
    IfxParameter myNewParameter = myParamCollection.Add(
      "EMPNO", IfxType.VarChar, 6);

    return myParm; }
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

"IfxParameterCollection.Add Method" on page 5-493

*IfxParameterCollection.Add (String, IfxType, Int32, String) Method:*

Adds a IfxParameter to the IfxCommand given the parameter name, data type, column width, and source column name.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overloads Public Function Add( _
   ByVal parameterName As String, _
   ByVal IfxType As IfxType
, _
   ByVal size As Integer, _
   ByVal sourceColumn As String _
) As IfxParameter

[C#]
public IfxParameter
 Add(
   string parameterName,
   IfxType
 IfxType,
   int size,
   string sourceColumn
);
[C++]
public: IfxParameter
* Add(
   String* parameterName,
   IfxType
 IfxType,
   int size,
```

```
   String* sourceColumn
);
[JScript]
public function Add(
   parameterName : String,
   IfxType : IfxType
,
   size : int,
   sourceColumn : String
) : IfxParameter
;
```

**Parameters**

**parameterName**
The name of the parameter.

**IfxType**
One of the IfxType values.

**size** The width of the column.

**sourceColumn**
The name of the source column.

**Return value**

The new IfxParameter object.

**Example**

[Visual Basic, C#] The following example creates a IfxParameterCollection , adds an instance of IfxParameter to the collection, and returns a reference to the new IfxParameter.

```
[Visual Basic]
Public Function CreateIfxParamColl(myConn As IfxConnection) As IfxParameter
  Dim myCommand As IfxCommand = New IfxCommand(
    "SELECT * FROM EMPLOYEE WHERE EMPNO = ?", myConn)
  Dim myParamCollection As IfxParameterCollection = myCommand.Parameters
  Dim myNewParameter As IfxParameter = myParamCollection.Add(
    "EMPNO", IfxType.Char, 6, "EMPNO")

  Return myParm
End Function 'CreateIfxParamColl

[C#]
public IfxParameter CreateIfxParamColl(IfxConnection myConn) {
    IfxCommand myCommand = new IfxCommand(
      "SELECT * FROM EMPLOYEE WHERE EMPNO = ?", myConn);
    IfxParameterCollection myParamCollection = myCommand.Parameters;
    IfxParameter myNewParameter = myParamCollection.Add(
      "EMPNO", IfxType.Char, 6, "EMPNO");

    return myParm;
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489
"IfxParameterCollection Members" on page 5-491
"IBM.Data.Informix Namespace" on page 5-1
"IfxParameterCollection.Add Method" on page 5-493

**IfxParameterCollection.AddRange Method:**

Appends an array of values to the IfxParameterCollection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Sub AddRange ( values As Array )
[C#]
public override void AddRange ( Array values )
[C++]
public:
virtual void AddRange ( Array^ values ) override
[JScript]
public override function AddRange ( values : Array )
```

**Parameters**

**values** The array of values to be appended to the IfxParameterCollection.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxParameterCollection Class" on page 5-489
"IfxParameterCollection Members" on page 5-491
"IBM.Data.Informix Namespace" on page 5-1

**IfxParameterCollection.Clear Method:**

Removes all items from the collection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Public Sub Clear() Implements IList.Clear
[C#]
public void Clear();
[C++]
public: __sealed void Clear();
[JScript]
public function Clear();
```

**Implements:**
IList.Clear

**Example**

[Visual Basic, C#] The following example creates a IfxParameterCollection, adds instances of IfxParameter to the collection, displays the names of its IfxParameter objects, and clears the collection.

```
[Visual Basic]
Public Sub CreateIfxParamColl(myCommand As IfxCommand)
    Dim myParamCollection As IfxParameterCollection = myCommand.Parameters
    myParamCollection.Add("EMPNO", IfxType.Char)
    myParamCollection.Add("PHOTO_FORMAT", IfxType.VarChar)
    myParamCollection.Add("PICTURE", IfxType.Blob)
    Dim myParamNames As String = ""
    Dim i As Integer
    For i = 0 To myParamCollection.Count - 1
        myParamNames += myParamCollection(i).ToString() + ControlChars.Cr
    Next i
    MessageBox.Show(myParamNames)
    myParamCollection.Clear()
    MessageBox.Show(myParamCollection.Count)
End Sub 'CreateIfxParamColl
```

```
[C#]
public void CreateIfxParamColl(IfxCommand myCommand)
{
    IfxParameterCollection myParamCollection = myCommand.Parameters;
    myParamCollection.Add("EMPNO", IfxType.Char);
    myParamCollection.Add("PHOTO_FORMAT", IfxType.VarChar);
    myParamCollection.Add("PICTURE", IfxType.Blob);
    string myParamNames = "";
    for (int i=0; i < myParamCollection.Count; i++)
       myParamNames += myParamCollection[i].ToString() + "\n";
    MessageBox.Show(myParamNames);
    myParamCollection.Clear();
    MessageBox.Show(myParamCollection.Count)
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

### IfxParameterCollection.Contains Method:

Gets a value indicating whether a IfxParameter object exists in the collection.

**Overload list**

| Name | Description |
|---|---|
| Contains(Object) As Boolean Implements IList.Contains | Gets a value indicating whether a IfxParameter object exists in the collection. |
| Contains(String) As Boolean Implements IDataParameterCollection.Contains | Gets a value indicating whether a IfxParameter object with the specified parameter name exists in the collection. |

**Example**

[Visual Basic, C#] The following example searches for a IfxParameter with a given IfxParameter.ParameterName within a IfxParameterCollection. If the parameter exists, the example displays the name and index of the parameter. If the parameter does not exist, the example displays an error. This example assumes that a IfxParameterCollection has already been created.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of Contains. For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub SearchIfxParams()
    ' ...
    ' create IfxParameterCollection myParameters
    ' ...
    If Not myParameters.Contains("Description") Then
        MessageBox.Show("ERROR: no such parameter in the collection")
    Else
        MessageBox.Show("Name: " & myParameters("Description").ToString() & _
                        "Index: " & myParameters.IndexOf("Description").ToString())
    End If
End Sub 'SearchIfxParams

[C#]
public void SearchIfxParams() {
    // ...
    // create IfxParameterCollection myParameters
    // ...
    if (!myParameters.Contains("Description"))
       MessageBox.Show("ERROR: no such parameter in the collection");
    else
       MessageBox.Show("Name: " + myParameters["Description"].ToString() +
                 "Index: " + myParameters.IndexOf("Description").ToString());
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

*IfxParameterCollection.Contains (Object) Method:*

Gets a value indicating whether a IfxParameter object exists in the collection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Overloads Public Function Contains( _
   ByVal value As Object _
) As Boolean Implements IList.Contains
[C#]
public bool Contains(
   object value
);
[C++]
public: __sealed bool Contains(
   Object* value
);
[JScript]
public function Contains(
   value : Object
) : Boolean;
```

**Implements:**
IList.Contains

**Parameters**

**value** The value of the IfxParameter object to find.

### Return value

`true` if the collection contains the IfxParameter ; otherwise, `false`.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentNullException | The **value** parameter is null. |
| InvalidCastException | The **value** parameter was not a IfxParameter . |

### Example

[Visual Basic, C#] The following example searches for a IfxParameter within a IfxParameterCollection . If the parameter exists, the example displays the index of the parameter. If the parameter does not exist, the example displays an error. This example assumes that a IfxParameterCollection has already been created.

```
[Visual Basic]
Public Sub SearchIfxParams()
    ' ...
    ' create IfxParameterCollection myParameters and IfxParameter myNewParam
    ' ...
    If Not myParameters.Contains(CType(myNewParam, Object)) Then
        MessageBox.Show("ERROR: no such parameter in the collection")
    Else
        MessageBox.Show("match on parameter #" &
        myParameters.IndexOf(CType(myNewParam, Object)).ToString())
    End If
End Sub 'SearchIfxParams

[C#]
public void SearchIfxParams() {
    // ...
    // create IfxParameterCollection myParameters and IfxParameter myNewParam
    // ...
    if (!myParameters.Contains((Object) myNewParam))
       MessageBox.Show("ERROR: no such parameter in the collection");
    else
       MessageBox.Show("match on parameter #" +
                 myParameters.IndexOf((Object) myNewParam).ToString());
}
```

#### Reference

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

"IfxParameterCollection.Contains Method" on page 5-503

*IfxParameterCollection.Contains (String) Method:*

Gets a value indicating whether a IfxParameter object with the specified parameter name exists in the collection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
NotOverridable Overloads Public Function Contains( _
   ByVal value As String _
) As Boolean Implements IDataParameterCollection.Contains
[C#]
public bool Contains(
   string value
);
[C++]
public: __sealed bool Contains(
   String* value
);
[JScript]
public function Contains(
   value : String
) : Boolean;
```

### Implements:

IDataParameterCollection.Contains

### Parameters

**value** The name of the IfxParameter object to find.

### Return value

`true` if the collection contains the parameter; otherwise, `false`.

### Example

[Visual Basic, C#] The following example searches for a IfxParameter with a given IfxParameter.ParameterName within a IfxParameterCollection. If the parameter exists, the example displays the name and index of the parameter. If the parameter does not exist, the example displays an error. This example assumes that a IfxParameterCollection has already been created.

```
[Visual Basic]
Public Sub SearchIfxParams()
    ' ...
    ' create IfxParameterCollection myParameters
    ' ...
    If Not myParameters.Contains("Description") Then
        MessageBox.Show("ERROR: no such parameter in the collection")
    Else
        MessageBox.Show("Name: " & myParameters("Description").ToString() & _
                        "Index: " & myParameters.IndexOf("Description").ToString())
    End If
End Sub 'SearchIfxParams

[C#]
public void SearchIfxParams() {
    // ...
    // create IfxParameterCollection myParameters
    // ...
    if (!myParameters.Contains("Description"))
       MessageBox.Show("ERROR: no such parameter in the collection");
    else
       MessageBox.Show("Name: " + myParameters["Description"].ToString() +
                  "Index: " + myParameters.IndexOf("Description").ToString());
 }
```

### Reference

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491
"IBM.Data.Informix Namespace" on page 5-1
"IfxParameterCollection.Contains Method" on page 5-503

### IfxParameterCollection.CopyTo Method:

Copies IfxParameter objects from the IfxParameterCollection to the specified array.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
NotOverridable Public Sub CopyTo( _
   ByVal array As Array, _
   ByVal index As Integer _
) Implements ICollection.CopyTo
[C#]
public void CopyTo(
   Array array,
   int index
);
[C++]
public: __sealed void CopyTo(
   Array* array,
   int index
);
[JScript]
public function CopyTo(
   array : Array,
   index : int
);
```

**Implements:**
ICollection.CopyTo

### Parameters

**array** The target array into which to copy the IfxParameter objects.

**index** The starting index of the target array.

### Example

[Visual Basic, C#] The following example exports the IfxParameterCollection to an array of IfxParameter objects, doubles the size of the array by using CopyTo and returns the collection. It then clears the collection, and returns `true` if the parameters are no longer persisting. This example assumes that a IfxParameterCollection has already been created.

```
[Visual Basic]
Public Function DoubleYourParams() As Boolean
    ' ...
    ' create IfxParameterCollection myParameters
    ' ...
    Dim myParamArray(2 * myParameters.Count - 1) As IfxParameter
    myParameters.CopyTo(myParamArray, 0)
    myParameters.CopyTo(myParamArray, myParameters.Count)
    myParameters.Clear()
    Return True
End Function 'DoubleYourParams
```

```
[C#]
public bool DoubleYourParams() {
    // ...
    // create IfxParameterCollection myParameters
    // ...
    IfxParameter[] myParamArray = new IfxParameter[(2*myParameters.Count) - 1];
    myParameters.CopyTo(myParamArray, 0);
    myParameters.CopyTo(myParamArray, myParameters.Count);
    myParameters.Clear();
    return true;
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameterCollection.GetEnumerator Method:**

Returns an enumerator that iterates through the IfxParameterCollection. This member supports the Microsoft .NET Framework infrastructure and is not intended to be used directly from your code.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Public Function GetEnumerator() As IEnumerator _
   Implements IEnumerable.GetEnumerator
[C#]
public IEnumerator GetEnumerator();
[C++]
public: __sealed IEnumerator* GetEnumerator();
[JScript]
public function GetEnumerator() : IEnumerator;
```

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameterCollection.GetParameter Method:**

Returns the DbParameter object with the specified name in the IfxParameterCollection.

*Table 5-25. Overload List*

| Method | Description |
|---|---|
| GetParameter(int) | Returns the DbParameter object at the specified index in the IfxParameterCollection. |
| GetParameter(string) | Returns the DbParameter object with the specified name in the IfxParameterCollection. |

**Reference**

"IfxParameterCollection Class" on page 5-489

"IBM.Data.Informix Namespace" on page 5-1

*IfxParameterCollection.GetParameter(int) Method:*

Returns the DbParameter object at the specified index in the IfxParameterCollection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function GetParameter (index As Integer) As DbParameter
[C#]
public override DbParameter GetParameter (int index)
[C++]
public:
virtual DbParameter^ GetParameter (int index) override
[JScript]
public override function GetParameter (index : int) : DbParameter
```

**Parameters**

**index** The position of the parameter in the list of parameters.

**Return value**

A DbParameter instance representing the requested parameter.

**Exceptions**

| Exception type | Condition |
|---|---|
| IndexOutOfRange | A parameter with the specified index does not exist in the collection. |

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxParameterCollection Class" on page 5-489

"IBM.Data.Informix Namespace" on page 5-1

*IfxParameterCollection.GetParameter(string) Method:*

Returns the DbParameter object with the specified name in the IfxParameterCollection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function GetParameter (parameterName As String) As DbParameter
[C#]
public override DbParameter GetParameter (string parameterName)
[C++]
public:
virtual DbParameter^ GetParameter (String^ parameterName) override
[JScript]
public override function GetParameter (parameterName : String) : DbParameter
```

**Parameters**

**parameterName**
The name of the parameter.

**Return value**

A DbParameter instance representing the requested parameter.

**Exceptions**

| Exception type | Condition |
|---|---|
| IndexOutOfRange | A parameter with the specified name does not exist in the collection. |

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxParameterCollection Class" on page 5-489
"IBM.Data.Informix Namespace" on page 5-1

**IfxParameterCollection.Insert Method:**

Inserts a IfxParameter into the collection at the specified index.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Public Sub Insert( _
   ByVal index As Integer, _
```

```
   ByVal value As Object _
)
[C#]
public void Insert(
   int index,
   object value
);
[C++]
public: __sealed void Insert(
   int index,
   Object* value
);
[JScript]
public function Insert(
   index : int,
   value : Object
);
```

**Parameters**

**index** The zero-based index where the parameter is to be inserted within the collection.

**value** The IfxParameter to add to the collection.

**Exceptions**

| Exception type | Condition |
|---|---|
| ArgumentNullException | The **value** parameter is null. |
| IndexOutOfRange | A parameter with the specified name does not exist in the collection. |
| InvalidCastException | The **value** parameter was not a IfxParameter . |

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameterCollection.RemoveAt Method:**

Removes the specified IfxParameter from the collection.

**Overload list**

| Name | Description |
|---|---|
| RemoveAt(Integer) Implements IList.RemoveAt | Removes the IfxParameter at the specified index from the collection. |
| RemoveAt(String) Implements IDataParameterCollection.RemoveAt | Removes the IfxParameter with the specified name from the collection. |

**Example**

[Visual Basic, C#] The following example searches for a IfxParameter with the specified IfxParameter.ParameterName within a IfxParameterCollection. If the parameter exists, it is removed. This example assumes that an IfxParameterCollection has already been created.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of RemoveAt. For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub SearchIfxParams()
    ' ...
    ' create IfxParameterCollection myParameters and IfxParameter myNewParam
    ' ...
    If myParameters.Contains("Description") Then
        myParameters.RemoveAt("Description")
    End If
End Sub 'SearchIfxParams

[C#]
public void SearchIfxParams() {
    // ...
    // create IfxParameterCollection myParameters and IfxParameter myNewParam
    // ...
    if (myParameters.Contains("Description"))
       myParameters.RemoveAt("Description");
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

*IfxParameterCollection.RemoveAt (Int32) Method:*

Removes the IfxParameter at the specified index from the collection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Overloads Public Sub RemoveAt( _
   ByVal index As Integer _
)
[C#]
public void RemoveAt(
   int index
);
[C++]
public: __sealed void RemoveAt(
   int index
);
[JScript]
public function RemoveAt(
   index : int
);
```

**Parameters**

**index** The zero-based index of the parameter to remove.

### Exceptions

| Exception type | Condition |
|---|---|
| IndexOutOfRangeException | A parameter with the specified name does not exist in the collection. |

### Example

[Visual Basic, C#] The following example searches for a IfxParameter at the specified index within a IfxParameterCollection . If the parameter exists, it is removed. This example assumes that a IfxParameterCollection has already been created.

```
[Visual Basic]
Public
Sub SearchIfxParams() '
    ... '
    create IfxParameterCollection myParameters and IfxParameter myNewParam "Param7"
    ' ...
    If myParameters.Contains(Param7) Then
        myParameters.RemoveAt(7)
    End If
End Sub 'SearchIfxParams

[C#]
public
void SearchIfxParams() { //
    ... //
    create IfxParameterCollection myParameters and IfxParameter myNewParam "Param7"
    // ...
    if (myParameters.Contains(Param7))
       myParameters.RemoveAt(7);
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

"IfxParameterCollection.RemoveAt Method" on page 5-511

*IfxParameterCollection.RemoveAt (String) Method:*

Removes the IfxParameter with the specified name from the collection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
NotOverridable Overloads Public Sub RemoveAt( _
   ByVal parameterName As String _
)
[C#]
public void RemoveAt(
   string parameterName
);
[C++]
public: __sealed void RemoveAt(
```

```
   String* parameterName
);
[JScript]
public function RemoveAt(
   parameterName : String
);
```

**Parameters**

**parameterName**
The name of the parameter to remove.

**Exceptions**

| Exception type | Condition |
|---|---|
| IndexOutOfRangeException | A parameter with the specified name does not exist in the collection. |

**Example**

[Visual Basic, C#] The following example searches for a IfxParameter with the specified IfxParameter.Name within a IfxParameterCollection. If the parameter exists, it is removed. This example assumes that a IfxParameterCollection has already been created.

```
[Visual Basic]
Public Sub SearchIfxParams()
    ' ...
    ' create IfxParameterCollection myParameters and IfxParameter myNewParam
    ' ...
    If myParameters.Contains("Description") Then
        myParameters.RemoveAt("Description")
    End If
End Sub 'SearchIfxParams

[C#]
public void SearchIfxParams() {
    // ...
    // create IfxParameterCollection myParameters and IfxParameter myNewParam
    // ...
    if (myParameters.Contains("Description"))
       myParameters.RemoveAt("Description");
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

"IfxParameterCollection.RemoveAt Method" on page 5-511

**IfxParameterCollection.Remove Method:**

Removes the specified IfxParameter from the collection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
NotOverridable Public Sub Remove( _
   ByVal value As Object _
)
[C#]
public void Remove(
   object value
);
[C++]
public: __sealed void Remove(
   Object* value
);
[JScript]
public function Remove(
   value : Object
);
```

### Parameters

**value** The IfxParameter object to remove from the collection.

### Exceptions

| Exception type | Condition |
|---|---|
| ArgumentException | The parameter does not exist. |
| InvalidCastException | The parameter is not a valid IfxParameter . |

### Example

[Visual Basic, C#] The following example searches for a IfxParameter object in a IfxParameterCollection . If the parameter exists, it is removed. This example assumes that a IfxParameterCollection has already been created.

```
[Visual Basic]
Public Sub SearchIfxParams()
    ' ...
    ' create IfxParameterCollection myParameters and IfxParameter myNewParam
    ' ...
    If myParameters.Contains(CType(myNewParam, Object)) Then
        myParameters.Remove(CType(myNewParam, Object))
    End If
End Sub 'SearchIfxParams

[C#]
public void SearchIfxParams() {
    // ...
    // create IfxParameterCollection myParameters and IfxParameter myNewParam
    // ...
    if (myParameters.Contains((Object) myNewParam))
       myParameters.Remove(((Object) myNewParam));
 }
```

### Reference

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

### IfxParameterCollection.SetParameter Method:

Sets the specified DbParameter to the specified value.

*Table 5-26. Overload List*

| Method | Description |
|---|---|
| SetParameter(int, System.Data.Common.DbParameter) | Sets the DbParameter at the specified index to the specified value. |
| SetParameter(string, System.Data.Common.DbParameter) | Sets the DbParameter with the specified name to the specified value. |

**Reference**

"IfxParameterCollection Class" on page 5-489

"IBM.Data.Informix Namespace" on page 5-1

*IfxParameterCollection.SetParameter(int, System.Data.Common.DbParameter) Method:*

Sets the DbParameter at the specified index to the specified value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Sub SetParameter (index As Integer, value As DbParameter)
[C#]
public override void SetParameter (int index, value DbParameter)
[C++]
public:
virtual void SetParameter (int index, value^ DbParameter) override
[JScript]
public override function SetParameter (index : int, value : DbParameter
```

**Parameters**

**index** The position of the parameter in the zero-based index.

**value** The DbParameter instance to be assigned at the specified index.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxParameterCollection Class" on page 5-489

"IBM.Data.Informix Namespace" on page 5-1

*IfxParameterCollection.SetParameter(string, System.Data.Common.DbParameter) Method:*

Sets the DbParameter with the specified name to the specified value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Overrides Sub SetParameter (parameterName As String, value As DbParameter)
[C#]
public override void SetParameter (string parameterName, value DbParameter)
[C++]
public:
virtual void SetParameter (String parameterName, value^ DbParameter) override
[JScript]
public override function SetParameter (parameterName : String, value : DbParameter
```

### Parameters

**parameterName**
A string representing the name of the parameter to be added.

**value** The DbParameter instance to be assigned at the specified index.

### Version information

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxParameterCollection Class" on page 5-489
"IBM.Data.Informix Namespace" on page 5-1

## IfxParameterCollection Properties

The properties of the IfxParameterCollection class are listed here. For a complete list of IfxParameterCollection class members, see the IfxParameterCollection Members topic.

*Table 5-27. Public Properties*

| Method | Description |
|---|---|
| Count | Gets the number of IfxParameter objects in the collection. |
| IsFixedSize | Gets a value that indicates whether the IfxParameterCollection has a fixed size. |
| IsReadOnly | Gets a value that indicates whether the IfxParameterCollection is read-only. |
| IsSynchronized | Gets a value that indicates whether the IfxParameterCollection is synchronized. |
| this | Overloaded. Gets or sets the IfxParameter with a specified attribute.<br><br>In C#, this property is the indexer for the IfxParameterCollection class. |
| SyncRoot | Gets an object that can be used to synchronize access to the IfxParameterCollection. |

**Reference**

"IfxParameterCollection Class" on page 5-489
"IBM.Data.Informix Namespace" on page 5-1

### IfxParameterCollection.Count Property:

Gets the number of IfxParameter objects in the collection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Count As Integer
[C#]
public int Count {get;}
[C++]
public: __property int get_Count();
[JScript]
public function get Count() : int;
```

**Property value**

The number of IfxParameter objects in the collection.

**Example**

[Visual Basic, C#] The following example creates a IfxParameterCollection, adds instances of IfxParameter to the collection, displays the names of the IfxParameter objects, and then clears the collection.

```
[Visual Basic]
Public Sub CreateIfxParamColl(myCommand As IfxCommand)
    Dim myParamCollection As IfxParameterCollection = myCommand.Parameters
    myParamCollection.Add("EMPNO", IfxType.Char)
    myParamCollection.Add("PHOTO_FORMAT", IfxType.VarChar)
    myParamCollection.Add("PICTURE", IfxType.Blob)
    Dim myParamNames As String = ""
    Dim i As Integer
    For i = 0 To myParamCollection.Count - 1
        myParamNames += myParamCollection(i).ToString() + ControlChars.Cr
    Next i
    MessageBox.Show(myParamNames)
    myParamCollection.Clear()
End Sub 'CreateIfxParamColl

[C#]
public void CreateIfxParamColl(IfxCommand myCommand) {
    IfxParameterCollection myParamCollection = myCommand.Parameters;
    myParamCollection.Add("EMPNO", IfxType.Char);
    myParamCollection.Add("PHOTO_FORMAT", IfxType.VarChar);
    myParamCollection.Add("PICTURE", IfxType.Blob);
    string myParamNames = "";
    for (int i=0; i < myParamCollection.Count; i++)
       myParamNames += myParamCollection[i].ToString() + "\n";
    MessageBox.Show(myParamNames);
    myParamCollection.Clear();
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameterCollection.IsFixedSize Property:**

Gets a value that indicates whether the IfxParameterCollection has a fixed size.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsFixedSize As Boolean
[C#]
public bool IsFixedSize {get;}
[C++]
public: __property bool get_IsFixedSize();
[JScript]
public final function get IsFixedSize() : boolean
```

**Property value**

`true` if IfxParameterCollection has a fixed size; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxParameterCollection Class" on page 5-489

"IBM.Data.Informix Namespace" on page 5-1

**IfxParameterCollection.IsReadOnly Property:**

Gets a value that indicates whether the IfxParameterCollection is read-only.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsReadOnly As Boolean
[C#]
public bool IsReadOnly {get;}
[C++]
public: __property bool get_IsReadOnly();
[JScript]
public final function get IsReadOnly() : boolean
```

**Property value**

`true` if IfxParameterCollection is read-only; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxParameterCollection Class" on page 5-489

"IBM.Data.Informix Namespace" on page 5-1

### IfxParameterCollection.IsSynchronized Property:

Gets a value that indicates whether the IfxParameterCollection is synchronized.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

#### Syntax

```
[Visual Basic]
Public ReadOnly Property IsSynchronized As Boolean
[C#]
public bool IsSynchronized {get;}
[C++]
public: __property bool get_IsSynchronized();
[JScript]
public final function get IsSynchronized() : boolean
```

#### Property value

`true` if IfxParameterCollection is synchronized; otherwise, `false`.

#### Version information

**.NET Framework version**
Supported in: 2.0 and 3.0

**Reference**

"IfxParameterCollection Class" on page 5-489
"IBM.Data.Informix Namespace" on page 5-1

### IfxParameterCollection.SyncRoot Property:

Gets an object that can be used to synchronize access to the IfxParameterCollection.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

#### Syntax

```
[Visual Basic]
Public ReadOnly Property SyncRoot As Object
[C#]
public Object SyncRoot {get;}
[C++]
public: __property Object get_SyncRoot();
[JScript]
public final function get SyncRoot() : Object
```

#### Property value

An Object that can be used to synchronize access to the IfxParameterCollection.

### Version information

**.NET Framework version**

Supported in: 2.0 and 3.0

**Reference**

"IfxParameterCollection Class" on page 5-489

"IBM.Data.Informix Namespace" on page 5-1

### IfxParameterCollection.this Property:

Gets or sets the IfxParameter with a specified attribute.

### Overload list

| Name | Description |
|---|---|
| this(Integer) As IfxParameter | Gets or sets the IfxParameter at the specified index. |
| this(String) As IfxParameter | Gets or sets the IfxParameter with the specified name. |

### Example

[Visual Basic, C#] The following example creates a IfxParameterCollection , adds instances of IfxParameter to the collection, displays the names of its IfxParameter objects, and then clears the collection.

**Note:** [Visual Basic, C#] This example shows how to use one of the overloaded versions of the this property (IfxParameterCollection indexer). For other examples that might be available, see the individual overload topics.

```
[Visual Basic]
Public Sub CreateIfxParamColl(myCommand As IfxCommand)
    Dim myParamCollection As IfxParameterCollection = myCommand.Parameters
    myParamCollection.Add("EMPNO", IfxType.Char)
    myParamCollection.Add("PHOTO_FORMAT", IfxType.VarChar)
    myParamCollection.Add("PICTURE", IfxType.Blob)
    Dim myParamNames As String = ""
    Dim i As Integer
    For i = 0 To myParamCollection.Count - 1
        myParamNames += myParamCollection(i).ToString() + ControlChars.Cr
    Next i
    MessageBox.Show(myParamNames)
    myParamCollection.Clear()
End Sub 'CreateIfxParamColl

[C#]
public void CreateIfxParamColl(IfxCommand myCommand) {
    IfxParameterCollection myParamCollection = myCommand.Parameters;
    myParamCollection.Add("EMPNO", IfxType.Char);
    myParamCollection.Add("PHOTO_FORMAT", IfxType.VarChar);
    myParamCollection.Add("PICTURE", IfxType.Blob);
    string myParamNames = "";
    for (int i=0; i < myParamCollection.Count; i++)
       myParamNames += myParamCollection[i].ToString() + "\n";
    MessageBox.Show(myParamNames);
    myParamCollection.Clear();
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491
"IBM.Data.Informix Namespace" on page 5-1

*IfxParameterCollection.This (String) Property:*

Gets or sets the IfxParameter with the specified name.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Overloads Public Default Property this( _
   ByVal parameterName As String _
) As IfxParameter

[C#]
public IfxParameter
 this[
   string parameterName
] {get; set;}
[C++]
public: __property IfxParameter
* get_this(
   String* parameterName
);
public: __property void set_this(
   String* parameterName,
   IfxParameter*
);
[JScript]
returnValue = IfxParameterCollectionObject.this(parameterName);
IfxParameterCollectionObject.this(parameterName) = returnValue;
-or-
returnValue = IfxParameterCollectionObject(parameterName);
IfxParameterCollectionObject(parameterName) = returnValue;
```

### Property value

The IfxParameter with the specified name.

### Exceptions

| Exception type | Condition |
|---|---|
| IndexOutOfRangeException | The name specified does not exist. |

### Example

[Visual Basic, C#] The following example searches for a IfxParameter with a given IfxParameter.Name within a IfxParameterCollection. If the parameter exists, the example displays the name and index of the parameter. If the parameter does not exist, the example displays an error. This example assumes that an IfxParameterCollection has already been created.

```
[Visual Basic]
Public Sub SearchIfxParams()
    ' ...
    ' create IfxParameterCollection myParameters
```

```
    ' ...
    If Not myParameters.Contains("Description") Then
        MessageBox.Show("ERROR: no such parameter in the collection")
    Else
        MessageBox.Show("Name: " & myParameters("Description").ToString() & _
                        "Index: " & myParameters.IndexOf("Description").ToString())
    End If
End Sub 'SearchIfxParams

[C#]
public void SearchIfxParams() {
    // ...
    // create IfxParameterCollection myParameters
    // ...
    if (!myParameters.Contains("Description"))
       MessageBox.Show("ERROR: no such parameter in the collection");
    else
       MessageBox.Show("Name: " + myParameters["Description"].ToString() +
                 "Index: " + myParameters.IndexOf("Description").ToString());
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

"IfxParameterCollection.this Property" on page 5-521

*IfxParameterCollection.this (Int32) Property:*

Gets or sets the IfxParameter at the specified index.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overloads Public Default Property this( _
   ByVal index As Integer _
) As IfxParameter

[C#]
public IfxParameter
 this[
   int index
] {get; set;}
[C++]
public: __property IfxParameter
* get_this(
   int index
);
public: __property void set_this(
   int index,
   IfxParameter*
);
[JScript]
returnValue = IfxParameterCollectionObject.this(index);
IfxParameterCollectionObject.this(index) = returnValue;
-or-
returnValue = IfxParameterCollectionObject(index);
IfxParameterCollectionObject(index) = returnValue;
```

**Property value**

The IfxParameter at the specified index.

**Exceptions**

| Exception type | Condition |
|---|---|
| IndexOutOfRangeException | The index specified does not exist. |

**Example**

[Visual Basic, C#] The following example creates a IfxParameterCollection, adds instances of IfxParameter to the collection, displays the names of its IfxParameter objects, and then clears the collection.

```
[Visual Basic]
Public Sub CreateIfxParamColl(myCommand As IfxCommand)
    Dim myParamCollection As IfxParameterCollection = myCommand.Parameters
    myParamCollection.Add("EMPNO", IfxType.Char)
    myParamCollection.Add("PHOTO_FORMAT", IfxType.VarChar)
    myParamCollection.Add("PICTURE", IfxType.Blob)
    Dim myParamNames As String = ""
    Dim i As Integer
    For i = 0 To myParamCollection.Count - 1
        myParamNames += myParamCollection(i).ToString() + ControlChars.Cr
    Next i
    MessageBox.Show(myParamNames)
    myParamCollection.Clear()
End Sub 'CreateIfxParamColl

[C#]
public void CreateIfxParamColl(IfxCommand myCommand) {
    IfxParameterCollection myParamCollection = myCommand.Parameters;
    myParamCollection.Add("EMPNO", IfxType.Char);
    myParamCollection.Add("PHOTO_FORMAT", IfxType.VarChar);
    myParamCollection.Add("PICTURE", IfxType.Blob);
    string myParamNames = "";
    for (int i=0; i < myParamCollection.Count; i++)
       myParamNames += myParamCollection[i].ToString() + "\n";
    MessageBox.Show(myParamNames);
    myParamCollection.Clear();
 }
```

**Reference**

"IfxParameterCollection Class" on page 5-489

"IfxParameterCollection Members" on page 5-491

"IBM.Data.Informix Namespace" on page 5-1

"IfxParameterCollection.this Property" on page 5-521

## IfxPermissionAttribute Class

Associates a security action with a custom security attribute.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## .NET Framework 2.0 3.0, 3.5, and 4.0 Inheritance hierarchy

System.Object
    System.Attribute
        System.Security.Permissions.SecurityAttribute
            System.Security.Permissions.CodeAccessSecurityAttribute
                System.Data.Common.DBDataPermissionAttribute
                    IBM.Data.Informix.IfxPermissionAttribute

## .NET Framework 2.0 3.0, 3.5, and 4.0 Syntax

```
[Visual Basic]
<AttributeUsage(AttributeTargets.Assembly Or AttributeTargets.Class _
   Or AttributeTargets.Struct Or AttributeTargets.Constructor Or _
   AttributeTargets.Method)>
<Serializable>
NotInheritable Public Class IfxPermissionAttribute
   Inherits DBDataPermissionAttribute
[C#]
[AttributeUsage(AttributeTargets.Assembly | AttributeTargets.Class
   | AttributeTargets.Struct | AttributeTargets.Constructor |
   AttributeTargets.Method)]
[Serializable]
public sealed class IfxPermissionAttribute :
   DBDataPermissionAttribute
[C++]
[AttributeUsage(AttributeTargets.Assembly | AttributeTargets.Class
   | AttributeTargets.Struct | AttributeTargets.Constructor |
   AttributeTargets.Method)]
[Serializable]
public __gc __sealed class IfxPermissionAttribute : public
   DBDataPermissionAttribute
[JScript]
public
   AttributeUsage(AttributeTargets.Assembly | AttributeTargets.Class |
   AttributeTargets.Struct | AttributeTargets.Constructor |
   AttributeTargets.Method)
   Serializable
class IfxPermissionAttribute extends
   DBDataPermissionAttribute
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxPermissionAttribute Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxPermissionAttribute Members

IfxPermissionAttribute overview

## Public Constructors

| Constructor | Description |
|---|---|
|  IfxPermissionAttribute | Initializes a new instance of the IfxPermissionAttribute class with one of the SecurityAction values. |

## Public Properties

| Property | Description |
|---|---|
|  Action (inherited from SecurityAttribute) | Gets or sets a security action. |
|  AllowBlankPassword (inherited from DBDataPermissionAttribute) | Gets a value indicating whether a blank password is allowed. |
|  TypeId (inherited from Attribute) | When implemented in a derived class, gets a unique identifier for this Attribute. |
|  Unrestricted (inherited from SecurityAttribute) | Gets or sets a value indicating whether full (unrestricted) permission to the resource protected by the attribute is declared. |

## Public Methods

| Method | Description |
|---|---|
|  CreatePermission | Overridden. Returns a IfxPermission object that is configured according to the attribute properties. |
|  Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
|  GetHashCode (inherited from Attribute) | Overridden. Returns the hash code for this instance. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  IsDefaultAttribute (inherited from Attribute) | When overridden in a derived class, returns an indication whether the value of this instance is the default value for the derived class. |
|  Match (inherited from Attribute) | When overridden in a derived class, returns a value indicating whether this instance equals a specified object. |
|  ToString (inherited from Object) | Returns a String that represents the current Object. |

### Protected Methods

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxPermissionAttribute Class" on page 5-524

"IBM.Data.Informix Namespace" on page 5-1

## IfxPermissionAttribute Constructor

Initializes a new instance of the IfxPermissionAttribute class with one of the SecurityAction values.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
<AttributeUsage(AttributeTargets.Assembly Or AttributeTargets.Class _
   Or AttributeTargets.Struct Or AttributeTargets.Constructor Or _
   AttributeTargets.Method)>
<Serializable>
Public Sub New( _
   ByVal action As SecurityAction _
)
[C#]
[AttributeUsage(AttributeTargets.Assembly | AttributeTargets.Class
   | AttributeTargets.Struct | AttributeTargets.Constructor |
   AttributeTargets.Method)]
[Serializable]
public IfxPermissionAttribute(
   SecurityAction action
);
[C++]
[AttributeUsage(AttributeTargets.Assembly | AttributeTargets.Class
   | AttributeTargets.Struct | AttributeTargets.Constructor |
   AttributeTargets.Method)]
[Serializable]
public: IfxPermissionAttribute(
   SecurityAction action
);
[JScript]
public
   AttributeUsage(AttributeTargets.Assembly | AttributeTargets.Class |
   AttributeTargets.Struct | AttributeTargets.Constructor |
   AttributeTargets.Method)
```

```
   Serializable
function IfxPermissionAttribute(
   action : SecurityAction
);
```

### Parameters

**action** One of the System.Security.Permissions.SecurityAction values representing an action that can be performed using declarative security.

### Return value

A IfxPermissionAttribute object.

**Reference**

"IfxPermissionAttribute Class" on page 5-524

"IfxPermissionAttribute Members" on page 5-525

"IBM.Data.Informix Namespace" on page 5-1

## IfxPermissionAttribute Methods

The methods of the IfxPermissionAttribute class are listed here. For a complete list of IfxPermissionAttribute class members, see the IfxPermissionAttribute Members topic.

### Public Methods

| Method | Description |
|---|---|
|  CreatePermission | Overridden. Returns a IfxPermission object that is configured according to the attribute properties. |
|  Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
|  GetHashCode (inherited from Attribute) | Overridden. Returns the hash code for this instance. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  IsDefaultAttribute (inherited from Attribute) | When overridden in a derived class, returns an indication whether the value of this instance is the default value for the derived class. |
|  Match (inherited from Attribute) | When overridden in a derived class, returns a value indicating whether this instance equals a specified object. |
|  ToString (inherited from Object) | Returns a String that represents the current Object. |

## Protected Methods

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxPermissionAttribute Class" on page 5-524

"IBM.Data.Informix Namespace" on page 5-1

**IfxPermissionAttribute.CreatePermission Method:**

Returns a IfxPermission object that is configured according to the attribute properties.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<AttributeUsage(AttributeTargets.Assembly Or AttributeTargets.Class _
   Or AttributeTargets.Struct Or AttributeTargets.Constructor Or _
   AttributeTargets.Method)>
<Serializable>
Overrides Public Function CreatePermission() As IPermission
[C#]
[AttributeUsage(AttributeTargets.Assembly | AttributeTargets.Class
   | AttributeTargets.Struct | AttributeTargets.Constructor |
   AttributeTargets.Method)]
[Serializable]
public override IPermission CreatePermission();
[C++]
[AttributeUsage(AttributeTargets.Assembly | AttributeTargets.Class
   | AttributeTargets.Struct | AttributeTargets.Constructor |
   AttributeTargets.Method)]
[Serializable]
public: IPermission* CreatePermission();
[JScript]
public
   AttributeUsage(AttributeTargets.Assembly | AttributeTargets.Class |
   AttributeTargets.Struct | AttributeTargets.Constructor |
   AttributeTargets.Method)
   Serializable
override function CreatePermission() : IPermission;
```

**Return value**

A IfxPermission object.

**Reference**

"IfxPermissionAttribute Class" on page 5-524

"IfxPermissionAttribute Members" on page 5-525

"IBM.Data.Informix Namespace" on page 5-1

# IfxPermission Class

Enables the IBM Data Server Provider for .NET to ensure that a user has a security level adequate to access a database.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## .NET Framework 2.0 3.0, 3.5, and 4.0 Inheritance hierarchy

System.Object
  System.Security.CodeAccessPermission
    System.Data.Common.DBDataPermission
      IBM.Data.Informix.IfxPermission

## .NET Framework 2.0 3.0, 3.5, and 4.0 Syntax

```
[Visual Basic]
<Serializable>
NotInheritable Public Class IfxPermission
   Inherits DBDataPermission
[C#]
[Serializable]
public sealed class IfxPermission : DBDataPermission
[C++]
[Serializable]
public __gc __sealed class IfxPermission : public
   DBDataPermission
[JScript]
public
   Serializable
class IfxPermission extends DBDataPermission
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxPermission Members" on page 5-531

"IBM.Data.Informix Namespace" on page 5-1

## IfxPermission Members

IfxPermission overview

### Public Constructors

| Constructor | Description |
|---|---|
|  IfxPermission | Initializes a new instance of the IfxPermission class. |

### Public Properties

| Property | Description |
|---|---|
|  AllowBlankPassword (inherited from DBDataPermission) | Gets a value indicating whether a blank password is allowed. |

### Public Methods

| Method | Description |
|---|---|
|  Add | Adds access for the specified connection string to the existing state of the IfxPermission. |
|  Assert (inherited from CodeAccessPermission) | Asserts that calling code can access the resource identified by the current permission through the code that calls this method, even if callers higher in the stack have not been granted permission to access the resource. |
|  Copy (inherited from DBDataPermission) | Overridden. Creates and returns an identical copy of the current permission object. |
|  Demand (inherited from CodeAccessPermission) | Forces a SecurityException at run time if all callers higher in the call stack have not been granted the permission specified by the current instance. |
|  Deny (inherited from CodeAccessPermission) | Prevents callers higher in the call stack from using the code that calls this method to access the resource specified by the current instance. |
|  Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
|  FromXml (inherited from DBDataPermission) | Overridden. Reconstructs a security object with a specified state from an XML encoding. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |

| Method | Description |
|---|---|
| Intersect (inherited from DBDataPermission) | Overridden. Returns a new permission object representing the intersection of the current permission object and the specified permission object. |
| IsSubsetOf (inherited from DBDataPermission) | Overridden. Returns a value indicating whether the current permission object is a subset of the specified permission object. |
| IsUnrestricted (inherited from DBDataPermission) | Returns a value indicating whether the permission can be represented as unrestricted without any knowledge of the permission semantics. |
| PermitOnly (inherited from CodeAccessPermission) | Prevents callers higher in the call stack from using the code that calls this method to access all resources except for the resource specified by the current instance. |
| ToString (inherited from CodeAccessPermission) | Overridden. Creates and returns a string representation of the current permission object. |
| ToXml (inherited from DBDataPermission) | Overridden. Creates an XML encoding of the security object and its current state. |
| Union (inherited from DBDataPermission) | Overridden. Returns a new permission object that is the union of the current and specified permission objects. |

### Protected Methods

| Method | Description |
|---|---|
| Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxPermission Class" on page 5-530

"IBM.Data.Informix Namespace" on page 5-1

## IfxPermission Constructor

Initializes a new instance of the IfxPermission class.

### Overload list

| Name | Description |
|---|---|
| New() | Initializes a new instance of the IfxPermission class. |

| Name | Description |
|---|---|
| New(PermissionState) | Initializes a new instance of the IfxPermission class with one of the PermissionState values. |

**Reference**

"IfxPermission Class" on page 5-530

"IfxPermission Members" on page 5-531

"IBM.Data.Informix Namespace" on page 5-1

**IfxPermission Constructor ():**

Initializes a new instance of the IfxPermission class.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Public Sub New()
[C#]
[Serializable]
public IfxPermission();
[C++]
[Serializable]
public: IfxPermission();
[JScript]
public
   Serializable
function IfxPermission();
```

**Remarks**

The base constructor initializes all fields to their default values.

**Version information**

**.NET Framework version**
Supported in:

**Reference**

"IfxPermission Class" on page 5-530

"IfxPermission Members" on page 5-531

"IBM.Data.Informix Namespace" on page 5-1

"IfxPermission Constructor" on page 5-532

**IfxPermission Constructor (PermissionState):**

Initializes a new instance of the IfxPermission class with one of the PermissionState values.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
<Serializable>
Public Sub New( _
   ByVal state As PermissionState _
)
[C#]
[Serializable]
public IfxPermission(
   PermissionState state
);
[C++]
[Serializable]
public: IfxPermission(
   PermissionState state
);
[JScript]
public
   Serializable
function IfxPermission(
   state : PermissionState
);
```

**Parameters**

**state** One of the System.Security.Permissions.PermissionState values.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxPermission Class" on page 5-530
"IfxPermission Members" on page 5-531
"IBM.Data.Informix Namespace" on page 5-1

## IfxPermission Methods

The methods of the IfxPermission class are listed here.

### Public Methods

| Method | Description |
|---|---|
| Add | Adds access for the specified connection string to the existing state of the IfxPermission. |
| Assert (inherited from CodeAccessPermission) | Asserts that calling code can access the resource identified by the current permission through the code that calls this method, even if callers higher in the stack have not been granted permission to access the resource. |
| Copy (inherited from DBDataPermission) | Overridden. Creates and returns an identical copy of the current permission object. |

| Method | Description |
| --- | --- |
|  Demand (inherited from CodeAccessPermission) | Forces a SecurityException at run time if all callers higher in the call stack have not been granted the permission specified by the current instance. |
|  Deny (inherited from CodeAccessPermission) | Prevents callers higher in the call stack from using the code that calls this method to access the resource specified by the current instance. |
|  Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
|  FromXml (inherited from DBDataPermission) | Overridden. Reconstructs a security object with a specified state from an XML encoding. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  Intersect (inherited from DBDataPermission) | Overridden. Returns a new permission object representing the intersection of the current permission object and the specified permission object. |
|  IsSubsetOf (inherited from DBDataPermission) | Overridden. Returns a value indicating whether the current permission object is a subset of the specified permission object. |
|  IsUnrestricted (inherited from DBDataPermission) | Returns a value indicating whether the permission can be represented as unrestricted without any knowledge of the permission semantics. |
|  PermitOnly (inherited from CodeAccessPermission) | Prevents callers higher in the call stack from using the code that calls this method to access all resources except for the resource specified by the current instance. |
|  ToString (inherited from CodeAccessPermission) | Overridden. Creates and returns a string representation of the current permission object. |
|  ToXml (inherited from DBDataPermission) | Overridden. Creates an XML encoding of the security object and its current state. |
|  Union (inherited from DBDataPermission) | Overridden. Returns a new permission object that is the union of the current and specified permission objects. |

## Protected Methods

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxPermission Members" on page 5-531

"IfxPermission Class" on page 5-530

"IBM.Data.Informix Namespace" on page 5-1

### IfxPermission.Add Method:

Adds access for the specified connection string to the existing state of the IfxPermission.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Sub Add ( _
        connectionString As String, _
        restrictions As String, _
        behavior As KeyRestrictionBehavior _
)
[C#]
public override void Add (
        string connectionString,
        string restrictions,
        KeyRestrictionBehavior behavior
)
[C++]
public:
virtual void Add (
        String^ connectionString,
        String^ restrictions,
        KeyRestrictionBehavior behavior
) override
[JScript]
public override function Add (
        connectionString As String,
        restrictions As String,
        behavior As KeyRestrictionBehavior
)
```

**Parameters**

**connectionString**
The connection string for the database connection.

**restrictions**
The key restrictions.

**behavior**
A KeyRestrictionBehavior enumeration.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxPermission Class" on page 5-530

"IBM.Data.Informix Namespace" on page 5-1

# IfxReal Structure

Represents the REAL Informix data type. Encapsulates the float .NET data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
Public Structure IfxReal
[C#]
public struct IfxReal
[C++]
public value class IfxReal
```

## Remarks

The IfxReal class is only supported for applications with connections to DB2 for Linux, UNIX, or Windows databases.

## Example

[C#] The following example demonstrates how to retrieve a single REAL column value from a table.

```
[C#]
  public static string getParam(IfxConnection conn)
  {
    string mySelectQuery = "SELECT REALCOL FROM TESTTBL";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);
    IfxDataReader reader = myCommand.ExecuteReader();

    if (reader.Read())
    {
      IfxReal selectValue = reader.GetIfxReal(0);

      if (!selectValue.IsNull) { return selectValue.ToString(); }
    }

    return "NULL";
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxReal Members

Represents the REAL Informix data type. Encapsulates the float .NET data type. The following tables list the members exposed by the IfxReal structure.

### Public Fields

| Field | Description |
|---|---|
| MaxValue | The maximum value for IfxReal: 3,402823e38. |
| MinValue | The minumum value for IfxReal: -3,402823e38. |
| Null | Null value for IfxReal. |
| Zero | Zero value for IfxReal. |

### Public Constructors

| Constructor | Description |
|---|---|
| IfxReal | Initializes a new IfxReal structure with the specified value. |

### Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxReal structure is null. |
| Value | Gets the value stored in the IfxReal structure. |

### Public Methods

| Method | Description |
|---|---|
| Parse | Converts the supplied String to IfxReal. |

| Method | Description |
|---|---|
|  ToString | Returns a string that represents the IfxReal structure. |
|  op_explicit | Converts the supplied IfxReal structure to a `float`. |
|  op_implicit | Converts the supplied `float` to IfxReal. |

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

## IfxReal Fields

The fields of the IfxReal class are listed here.

### Public Fields

| Field | Description |
|---|---|
|  MaxValue | The maximum value for IfxReal: 3,402823e38. |
|  MinValue | The minumum value for IfxReal: -3,402823e38. |
|  Null | Null value for IfxReal. |
|  Zero | Zero value for IfxReal. |

**Reference**

"IfxReal Members" on page 5-538

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

**IfxReal.MaxValue Field:**

The maximum value for IfxReal: 3,402823e38.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MaxValue As IfxReal
[C#]
public static readonly IfxReal MaxValue
[C++]
public:
static initonly IfxReal MaxValue
[JScript]
public static final var MaxValue () : IfxReal
```

**Remarks**

The value of this constant is `3,402823e38`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

**IfxReal.MinValue Field:**

The minumum value for IfxReal: -3,402823e38.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MinValue As IfxReal
[C#]
public static readonly IfxReal MinValue
[C++]
public:
static initonly IfxReal MinValue
[JScript]
public static final var MinValue () : IfxReal
```

**Remarks**

The value of this constant is `-3,402823e38`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

**IfxReal.Null Field:**

Null value for IfxReal.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxReal
[C#]
```

```
public static readonly IfxReal Null
[C++]
public:
static initonly IfxReal Null
[JScript]
public static final var Null () : IfxReal
```

**Remarks**

The value of this constant is `NULL`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

**IfxReal.Zero Field:**

Zero value for IfxReal.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Zero As IfxReal
[C#]
public static readonly IfxReal Zero
[C++]
public:
static initonly IfxReal Zero
[JScript]
public static final var Zero () : IfxReal
```

**Remarks**

The value of this constant is `0`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

## IfxReal Constructor

Initializes a new IfxReal object with the specified value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New(value as Float)
[C#]
public IfxReal(float value);
[C++]
public: IfxReal(float value);
[JScript]
public function IfxReal(value : float);
```

### Parameters

**value** A `float` value to populate the IfxReal instance.

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

## IfxReal Methods

The methods of the IfxReal structure are listed here.

### Public Methods

| Method | Description |
|---|---|
|  Parse | Converts the supplied String to IfxReal. |
|  ToString | Returns a string that represents the IfxReal structure. |
|  op_explicit | Converts the supplied IfxReal structure to a `float`. |
|  op_implicit | Converts the supplied `float` to IfxReal. |

**Reference**

"IfxReal Members" on page 5-538

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

**IfxReal.op_explicit Method:**

Converts the supplied IfxReal structure to a `float`.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Narrowing Operator CType (source As IfxReal) As Float
[C#]
```

```
public static implicit operator float (IfxReal source)
[C++]
public:
static implicit operator float (IfxReal source)
```

**Parameters**

**source** A IfxReal structure to be converted to a `float`.

**Return value**

A `float` value converted from a IfxReal instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

**IfxReal.op_implicit Method:**

Converts the supplied `float` to IfxReal.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Widening Operator CType (source As Float) As IfxReal
[C#]
public static implicit operator IfxReal (float source)
[C++]
public:
static implicit operator IfxReal (float source)
```

**Parameters**

**source** A `float` value to be converted to IfxReal.

**Return value**

A IfxReal structure with the value of **source**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

**IfxReal.Parse Method:**

Converts the supplied String to IfxReal.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Function Parse (szReal As String) As IfxReal
[C#]
public static IfxReal Parse (string szReal)
[C++]
public:
static IfxReal Parse (string szReal)
[JScript]
public static function Parse (szReal String ) : IfxReal
```

**Parameters**

**szReal** A String value to be converted to IfxReal.

**Return value**

A IfxReal structure with the numeric value of **szReal**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

**IfxReal.ToString Method:**

Returns a string that represents the IfxReal structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxReal structure.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

## IfxReal Properties

The properties of the IfxReal structure are listed here.

### Public Properties

| Property | Description |
|---|---|
|  IsNull | Gets a value that indicates if the value stored in the IfxReal structure is null. |
|  Value | Gets the value stored in the IfxReal structure. |

**Reference**

"IfxReal Members" on page 5-538

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

**IfxReal.IsNull Property:**

Gets a value that indicates if the value stored in the IfxReal structure is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

**IfxReal.Value Property:**

Gets the value stored in the IfxReal object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Value As Float
[C#]
public float Value {get;}
[C++]
public: __property float get_Value();
[JScript]
public function get Value() : float;
```

**Property value**

A `float` representing the IfxReal instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

# IfxRowId Structure

Represents the ROWID Informix data type. Encapsulates the byte [ ] .NET data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
Public Structure IfxRowId
[C#]
public struct IfxRowId
[C++]
public value class IfxRowId
```

## Remarks

The IfxRowId class is only supported for applications with connections to DB2 for z/OS databases or to DB2 for i5/OS databases. IfxRowId structures can only be

instantiated by reading ROWID columns from a database (by using the GetIfxRowId method from the IfxDataReader, IfxResultSet, IfxRecord, or IfxUpdatableRecord classes).

## Example

[C#] The following example demonstrates how to use the IfxRowId structure.

```
[C#]
  public static string getParam(IfxConnection conn, IfxRowId rowid)
  {
    string mySelectQuery = "SELECT EMPNO FROM EMPLOYEE WHERE EMP_ROWID=ROWID(?)";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);
    IfxParameter p1 = new IfxParameter();
    p1.Value = rowid;
    myCommand.Parameters.Add(p1);

    IfxDataReader reader = myCommand.ExecuteReader();

    if (reader.Read())
    {
      IfxString selectValue = reader.GetIfxString(0);

      if (!selectValue.IsNull) { return selectValue.ToString(); }
    }

    return "NULL";
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxRowId Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxRowId Members

Represents the ROWID Informix data type. Encapsulates the byte [ ] .NET data type. The following tables list the members exposed by the IfxRowId structure.

### Public Fields

| Field | Description |
|---|---|
| Null | Null value for IfxRowId. |

### Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxRowId structure is null. |

| Property | Description |
|---|---|
|  Value | Gets the value stored in the IfxRowId structure. |

### Public Methods

| Method | Description |
|---|---|
|  op_explicit | Converts the supplied IfxRowId structure to a byte array. |
|  ToString | Returns a string that represents the IfxRowId structure. |

**Reference**

"IfxRowId Structure" on page 5-546

"IBM.Data.Informix Namespace" on page 5-1

## IfxRowId Fields

The fields of the IfxRowId class are listed here.

### Public Fields

| Field | Description |
|---|---|
|  Null | Null value for IfxRowId. |

**Reference**

"IfxRowId Members" on page 5-547

"IfxRowId Structure" on page 5-546

"IBM.Data.Informix Namespace" on page 5-1

**IfxRowId.Null Field:**

Null value for IfxRowId.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxRowId
[C#]
public static readonly IfxRowId Null
[C++]
public:
static initonly IfxRowId Null
[JScript]
public static final var Null () : IfxRowId
```

**Remarks**

The value of this constant is `NULL`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxRowId Structure" on page 5-546

"IBM.Data.Informix Namespace" on page 5-1

## IfxRowId Methods

The methods of the IfxRowId structure are listed here.

### Public Methods

| Method | Description |
|---|---|
|  op_explicit | Converts the supplied IfxRowId structure to a byte array. |
|  ToString | Returns a string that represents the IfxRowId structure. |

**Reference**

"IfxRowId Members" on page 5-547

"IfxRowId Structure" on page 5-546

"IBM.Data.Informix Namespace" on page 5-1

**IfxRowId.op_explicit Method:**

Converts the supplied IfxRowId structure to a byte array.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Narrowing Operator CType (source As IfxRowId) As Byte()
[C#]
public static implicit operator byte[] (IfxRowId source)
[C++]
public:
static implicit operator array<unsigned char> (IfxRowId source)
```

**Parameters**

**source** A IfxRowId structure to be converted to a byte array.

**Return value**

A byte array converted from a IfxRowId instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxRowId Structure" on page 5-546

"IBM.Data.Informix Namespace" on page 5-1

**IfxRowId.ToString Method:**

Returns a string that represents the IfxRowId structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxRowId structure.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxRowId Structure" on page 5-546
"IBM.Data.Informix Namespace" on page 5-1

## IfxRowId Properties

The properties of the IfxRowId class are listed here.

### Public Properties

| Property | Description |
|---|---|
|  IsNull | Gets a value that indicates if the value stored in the IfxRowId structure is null. |
|  Value | Gets the value stored in the IfxRowId structure. |

**Reference**

"IfxRowId Members" on page 5-547
"IfxRowId Structure" on page 5-546
"IBM.Data.Informix Namespace" on page 5-1

**IfxRowId.IsNull Property:**

Gets a value that indicates if the value stored in the IfxRowId structure is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxRowId Structure" on page 5-546

"IBM.Data.Informix Namespace" on page 5-1

**IfxRowId.Value Property:**

Gets the value stored in the IfxRowId object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Value As Byte()
[C#]
public byte[] Value {get;}
[C++]
public: __property array<unsigned char> get_Value();
[JScript]
public function get Value() : byte[];
```

**Property value**

A byte array representing the IfxRowId instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxRowId Structure" on page 5-546

"IBM.Data.Informix Namespace" on page 5-1

# IfxRowUpdatedEventArgs Class

Provides data for the RowUpdated event.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
  System.EventArgs
    System.Data.Common.RowUpdatedEventArgs
      IBM.Data.Informix.IfxRowUpdatedEventArgs

## Syntax

```
[Visual Basic]
NotInheritable Public Class IfxRowUpdatedEventArgs
   Inherits RowUpdatedEventArgs
[C#]
public sealed class IfxRowUpdatedEventArgs : RowUpdatedEventArgs
[C++]
public __gc __sealed class IfxRowUpdatedEventArgs : public
   RowUpdatedEventArgs
[JScript]
public class IfxRowUpdatedEventArgs extends RowUpdatedEventArgs
```

## Remarks

The IfxDataAdapter.RowUpdated event is raised when an update to a row is completed.

When using the Update method, there are two events that occur for each data row updated: IfxDataAdapter.RowUpdating and RowUpdated. The order of execution for an Update is as follows:

1. The values in the DataRow are moved to the parameter values.
2. The IfxDataAdapter.RowUpdating event is raised.
3. The command executes.
4. If UpdateRowSource is set to FirstReturnedRecord, then the first returned result is placed in the DataRow.
5. If there are output parameters, they are placed in the DataRow.
6. The IfxDataAdapter.RowUpdated event is raised.
7. AcceptChanges is called.

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxRowUpdatedEventArgs Members"
"IBM.Data.Informix Namespace" on page 5-1

## IfxRowUpdatedEventArgs Members

IfxRowUpdatedEventArgs overview

### Public Constructors

| Constructor | Description |
|---|---|
| IfxRowUpdatedEventArgs | Initializes a new instance of the IfxRowUpdatedEventArgs class. |

### Public Properties

| Property | Description |
|---|---|
| Command | Gets the IfxCommand executed when Update is called. |
| IfxErrors | If any IfxException instances were thrown in the batch of commands, this property will reference a IfxErrorCollection instance. |
| Errors (inherited from RowUpdatedEventArgs) | Gets any errors generated by the IBM Data Server Provider for .NET when the Command was executed. |
| RecordsAffected (inherited from RowUpdatedEventArgs) | Gets the number of rows changed, inserted, or deleted by execution of the SQL statement. |
| Row (inherited from RowUpdatedEventArgs) | Gets the DataRow sent through an Update. |
| RowCount (inherited from RowUpdatedEventArgs) | The number of rows processed in the batch of commands sent to the database server. |
| StatementType (inherited from RowUpdatedEventArgs) | Gets the type of SQL statement executed. |
| Status (inherited from RowUpdatedEventArgs) | Gets the UpdateStatus of the Command. |
| TableMapping (inherited from RowUpdatedEventArgs) | Gets the DataTableMapping sent through an Update. |

### Public Methods

| Method | Description |
|---|---|
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |

| Method | Description |
|---|---|
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  ToString (inherited from Object) | Returns a String that represents the current Object. |

### Protected Methods

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxRowUpdatedEventArgs Class" on page 5-552

"IBM.Data.Informix Namespace" on page 5-1

## IfxRowUpdatedEventArgs Constructor

Initializes a new instance of the IfxRowUpdatedEventArgs class.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New( _
   ByVal row As DataRow, _
   ByVal command As IDbCommand, _
   ByVal statementType As StatementType, _
   ByVal tableMapping As DataTableMapping _
)
[C#]
public IfxRowUpdatedEventArgs(
   DataRow row,
   IDbCommand command,
   StatementType statementType,
   DataTableMapping tableMapping
);
[C++]
public: IfxRowUpdatedEventArgs(
   DataRow* row,
   IDbCommand* command,
   StatementType statementType,
   DataTableMapping* tableMapping
);
[JScript]
public function IfxRowUpdatedEventArgs(
```

```
   row : DataRow,
   command : IDbCommand,
   statementType : StatementType,
   tableMapping : DataTableMapping
);
```

### Parameters

**row** The DataRow sent through an update operation.

**command**
The IfxCommand executed when Update is called.

**statementType**
One of the System.Data.StatementType values that specifies the type of query executed.

**tableMapping**
The System.Data.Common.DataTableMapping sent through Update.

**Reference**

"IfxRowUpdatedEventArgs Class" on page 5-552

"IfxRowUpdatedEventArgs Members" on page 5-553

"IBM.Data.Informix Namespace" on page 5-1

## IfxRowUpdatedEventArgs Properties

The properties of the IfxRowUpdatedEventArgs class are listed here. For a complete list of IfxRowUpdatedEventArgs class members, see the IfxRowUpdatedEventArgs Members topic.

### Public Properties

| Property | Description |
|---|---|
| Command | Gets the IfxCommand executed when Update is called. |
| IfxErrors | If any IfxException instances were thrown in the batch of commands, this property will reference a IfxErrorCollection instance. |
| Errors (inherited from RowUpdatedEventArgs) | Gets any errors generated by the IBM Data Server Provider for .NET when the Command was executed. |
| RecordsAffected (inherited from RowUpdatedEventArgs) | Gets the number of rows changed, inserted, or deleted by execution of the SQL statement. |
| Row (inherited from RowUpdatedEventArgs) | Gets the DataRow sent through an Update. |
| RowCount (inherited from RowUpdatedEventArgs) | The number of rows processed in the batch of commands sent to the database server. |
| StatementType (inherited from RowUpdatedEventArgs) | Gets the type of SQL statement executed. |
| Status (inherited from RowUpdatedEventArgs) | Gets the UpdateStatus of the Command. |

| Property | Description |
|---|---|
|  TableMapping (inherited from RowUpdatedEventArgs) | Gets the DataTableMapping sent through an Update. |

**Reference**

"IfxRowUpdatedEventArgs Class" on page 5-552

"IBM.Data.Informix Namespace" on page 5-1

**IfxRowUpdatedEventArgs.Command Property:**

Gets the IfxCommand executed when Update is called.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Command As IfxCommand

[C#]
public new IfxCommand
 Command {get;}
[C++]
public: __property IfxCommand
* get_Command();
[JScript]
public function get Command() : IfxCommand
;
```

**Property value**

The IfxCommand executed when Update is called.

**Reference**

"IfxRowUpdatedEventArgs Class" on page 5-552

"IfxRowUpdatedEventArgs Members" on page 5-553

"IBM.Data.Informix Namespace" on page 5-1

**IfxRowUpdatedEventArgs.IfxErrors Property:**

If any IfxException instances were thrown in the batch of commands, this property will reference a collection of IfxError objects.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IfxErrors As IfxErrorCollection
[C#]
public IfxErrorCollection IfxErrors {get;}
[C++]
```

```
public: __property IfxErrorCollection get_IfxErrors();
[JScript]
public function get IfxErrors() : IfxErrorCollection;
```

**Property value**

If any IfxException instances were thrown in the batch of commands, this property will reference a IfxErrorCollection object. If no IfxException instances were thrown, this property will be `null`.

**Version information**

**.NET Framework version**

Supported in: 2.0 and 3.0

**Reference**

"IfxRowUpdatedEventArgs Class" on page 5-552

"IfxRowUpdatedEventArgs Members" on page 5-553

"IBM.Data.Informix Namespace" on page 5-1

# IfxRowUpdatedEventHandler Delegate

Represents the method that will handle the RowUpdated event of a IfxDataAdapter.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
<Serializable>
Public Delegate Sub IfxRowUpdatedEventHandler( _
   ByVal sender As Object, _
   ByVal e As IfxRowUpdatedEventArgs _
)
[C#]
[Serializable]
public delegate void IfxRowUpdatedEventHandler(
   object sender,
   IfxRowUpdatedEventArgs e
);
[C++]
[Serializable]
public __gc __delegate void IfxRowUpdatedEventHandler(
   Object* sender,
   IfxRowUpdatedEventArgs* e
);
```

## Remarks

The handler is not required perform any action, and your code should avoid generating exceptions or allowing exceptions to propagate to the calling method. Any exceptions that do reach the caller are ignored.

When you create a IfxRowUpdatedEventArgs delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event

handler delegates, see "Events and Delegates" in the .NET Framework SDK documentation.

**Reference**

"IBM.Data.Informix Namespace" on page 5-1

# IfxRowUpdatingEventArgs Class

Provides data for the RowUpdating event.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
  System.EventArgs
    System.Data.Common.RowUpdatingEventArgs
      IBM.Data.Informix.IfxRowUpdatingEventArgs

## Syntax

```
[Visual Basic]
NotInheritable Public Class IfxRowUpdatingEventArgs
   Inherits RowUpdatingEventArgs
[C#]
public sealed class IfxRowUpdatingEventArgs : RowUpdatingEventArgs
[C++]
public __gc __sealed class IfxRowUpdatingEventArgs : public
   RowUpdatingEventArgs
[JScript]
public class IfxRowUpdatingEventArgs extends RowUpdatingEventArgs
```

## Remarks

The RowUpdating event is raised before an update to a row.

When using the Update method, there are two events that occur for each data row updated: RowUpdating and RowUpdated. The order of execution for an update operation is as follows:

1. The values in the DataRow are moved to the parameter values.
2. The RowUpdating event is raised.
3. The command executes.
4. If UpdateRowSource is set to FirstReturnedRecord, the first returned result is placed in the DataRow.
5. If there are output parameters, they are placed in the DataRow.
6. The RowUpdated event is raised.
7. AcceptChanges is called.

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxRowUpdatingEventArgs Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxRowUpdatingEventArgs Members

IfxRowUpdatingEventArgs overview

*Table 5-28. Public Constructors*

| Constructor | Description |
|---|---|
| IfxRowUpdatingEventArgs | Initializes a new instance of the IfxRowUpdatingEventArgs class. |

*Table 5-29. Public Properties*

| Property | Description |
|---|---|
| Command | Gets or sets the IfxCommand to execute when Update is called. |
| Errors (inherited from RowUpdatingEventArgs) | Gets any errors generated by the IBM Data Server Provider for .NET when the Command executes. |
| Row (inherited from RowUpdatingEventArgs) | Gets the DataRow to send through an Update. |
| StatementType (inherited from RowUpdatingEventArgs) | Gets the type of SQL statement to execute. |
| Status (inherited from RowUpdatingEventArgs) | Gets the UpdateStatus of the Command. |
| TableMapping (inherited from RowUpdatingEventArgs) | Gets the DataTableMapping to send through the Update. |

*Table 5-30. Public Methods*

| Method | Description |
|---|---|
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

*Table 5-31. Protected Methods*

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxRowUpdatingEventArgs Class" on page 5-558

"IBM.Data.Informix Namespace" on page 5-1

## IfxRowUpdatingEventArgs Constructor

Initializes a new instance of the IfxRowUpdatingEventArgs class.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New( _
   ByVal row As DataRow, _
   ByVal command As IDbCommand, _
   ByVal statementType As StatementType, _
   ByVal tableMapping As DataTableMapping _
)
[C#]
public IfxRowUpdatingEventArgs(
   DataRow row,
   IDbCommand command,
   StatementType statementType,
   DataTableMapping tableMapping
);
[C++]
public: IfxRowUpdatingEventArgs(
   DataRow* row,
   IDbCommand* command,
   StatementType statementType,
   DataTableMapping* tableMapping
);
[JScript]
public function IfxRowUpdatingEventArgs(
   row : DataRow,
   command : IDbCommand,
   statementType : StatementType,
   tableMapping : DataTableMapping
);
```

### Parameters

**row** The DataRow to update.

**command**
The IfxCommand to execute during the update operation.

**statementType**
One of the System.Data.StatementType values that specifies the type of query executed.

**tableMapping**
The DataTableMapping sent through Update.

**Reference**

"IfxRowUpdatingEventArgs Class" on page 5-558
"IfxRowUpdatingEventArgs Members" on page 5-559
"IBM.Data.Informix Namespace" on page 5-1

## IfxRowUpdatingEventArgs Properties

The properties of the IfxRowUpdatingEventArgs class are listed here. For a complete list of IfxRowUpdatingEventArgs class members, see the IfxRowUpdatingEventArgs Members topic.

*Table 5-32. Public Properties*

| Property | Description |
|---|---|
| Command | Gets or sets the IfxCommand to execute when Update is called. |
| Errors (inherited from RowUpdatingEventArgs) | Gets any errors generated by the IBM Data Server Provider for .NET when the Command executes. |
| Row (inherited from RowUpdatingEventArgs) | Gets the DataRow to send through an Update. |
| StatementType (inherited from RowUpdatingEventArgs) | Gets the type of SQL statement to execute. |
| Status (inherited from RowUpdatingEventArgs) | Gets the UpdateStatus of the Command. |
| TableMapping (inherited from RowUpdatingEventArgs) | Gets the DataTableMapping to send through the Update. |

**Reference**

"IfxRowUpdatingEventArgs Class" on page 5-558
"IBM.Data.Informix Namespace" on page 5-1

**IfxRowUpdatingEventArgs.Command Property:**

Gets or sets the IfxCommand to execute when Update is called.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Property Command As IfxCommand

[C#]
public new IfxCommand
 Command {get; set;}
[C++]
public: __property IfxCommand
* get_Command();
public: __property void set_Command(IfxCommand
*);
[JScript]
public function get Command() : IfxCommand
;
public function set Command(IfxCommand
);
```

### Property value

The IfxCommand to execute when Update is called.

**Reference**

"IfxRowUpdatingEventArgs Class" on page 5-558

"IfxRowUpdatingEventArgs Members" on page 5-559

"IBM.Data.Informix Namespace" on page 5-1

## IfxRowUpdatingEventHandler Delegate

Represents the method that will handle the RowUpdating event of a IfxDataAdapter.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
<Serializable>
Public Delegate Sub IfxRowUpdatingEventHandler( _
   ByVal sender As Object, _
   ByVal e As IfxRowUpdatingEventArgs _
)
[C#]
[Serializable]
public delegate void IfxRowUpdatingEventHandler(
   object sender,
   IfxRowUpdatingEventArgs e
);
[C++]
[Serializable]
public __gc __delegate void IfxRowUpdatingEventHandler(
   Object* sender,
   IfxRowUpdatingEventArgs* e
);
```

### Remarks

The handler is not required perform any action, and your code should avoid generating exceptions or allowing exceptions to propagate to the calling method. Any exceptions that do reach the caller are ignored.

Use the handler to process IfxRowUpdatingEventArgs and to specify how updates are processed. For example, you may opt to skip the update of the current row or skip the update of all remaining rows. Note that the rows are updated in the order that they were received from the database.

When you create a IfxRowUpdatingEventArgs delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see "Events and Delegates" in the .NET Framework SDK documentation.

**Reference**

"IBM.Data.Informix Namespace" on page 5-1

## IfxRowsCopiedEventArgs Class

Provides data for the IfxRowsCopied event.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Inheritance hierarchy

System.Object
  System.EventArgs
    IBM.Data.Informix.IfxRowsCopiedEventArgs

### Syntax

```
[Visual Basic]
Public Class IfxRowsCopiedEventArgs
         Inherits EventArgs
[C#]
public class IfxRowsCopiedEventArgs : EventArgs
[C++]
public ref class IfxRowsCopiedEventArgs : public EventArgs
[JScript]
public class IfxRowsCopiedEventArgs extends EventArgs
```

### Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxRowsCopiedEventArgs Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxRowsCopiedEventArgs Members

Provides data for the IfxRowsCopied event. The following tables list the members exposed by the IfxRowsCopiedEventArgs class.

### Public Properties

| Name | Description |
|---|---|
|  Abort | Gets or sets a boolean value that determines whether the current IfxBulkCopy operation will be aborted or not. |
|  RowsCopied | Gets a value indicating the number of rows copied during the current bulk copy operation. |

**Reference**

"IfxRowsCopiedEventArgs Class" on page 5-563

"IBM.Data.Informix Namespace" on page 5-1

## IfxRowsCopiedEventArgs Properties

The properties of the IfxRowsCopiedEventArgs class are listed here.

### Public Properties

| Name | Description |
|---|---|
|  Abort | Gets or sets a boolean value that determines whether the current IfxBulkCopy operation will be aborted or not. |
|  RowsCopied | Gets a value indicating the number of rows copied during the current bulk copy operation. |

**Reference**

"IfxRowsCopiedEventArgs Members"

"IfxRowsCopiedEventArgs Class" on page 5-563

"IBM.Data.Informix Namespace" on page 5-1

**IfxRowsCopiedEventArgs.Abort Property:**

Gets or sets a boolean value that determines whether the current IfxBulkCopy operation will be aborted or not.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Property Abort As Boolean
[C#]
public bool Abort {get; set;}
[C++]
public: __property bool get_Abort();
```

```
public: __property void set_Abort(bool);
[JScript]
public function get Abort() : boolean;
public function set Abort(boolean);
```

**Property value**

If Abort is `true`, the current IfxBulkCopy operation will be aborted. If this value is `false`, the operation will continue.

**Remarks**

You can use the Abort property to cancel a IfxBulkCopy operation. For bulk copy operations on DB2 for Linux, UNIX, and Windows databases, the rows that have been copied and committed cannot be rolled back.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxRowsCopiedEventArgs Class" on page 5-563

"IBM.Data.Informix Namespace" on page 5-1

**IfxRowsCopiedEventArgs.RowsCopied Property:**

Gets a value indicating the number of rows copied during the current bulk copy operation.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property RowsCopied As Long
[C#]
public long RowsCopied { get; }
[C++]
public: __property long get_RowsCopied();
[JScript]
public function get RowsCopied() : long;
```

**Property value**

A long integer indicating the number of rows copied.

**Remarks**

This value assumes a new value each time a IfxBulkCopy.WriteToServer method is called.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5 and 4.0

**Reference**

"IfxRowsCopiedEventArgs Class" on page 5-563
"IBM.Data.Informix Namespace" on page 5-1

# IfxRowsCopiedEventHandler Delegate

Represents the method that will handle the RowsCopied event of a IfxBulkCopy operation.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
<Serializable>
Public Delegate Sub IfxRowsCopiedEventHandler( _
   ByVal sender As Object, _
   ByVal e As IfxRowsCopiedEventArgs _
)
[C#]
[Serializable]
public delegate void IfxRowsCopiedEventHandler(
   object sender,
   IfxRowsCopiedEventArgs e
);
[C++]
[Serializable]
public __gc __delegate void IfxRowsCopiedEventHandler(
   Object* sender,
   IfxRowsCopiedEventArgs* e
);
```

## Remarks

The handler is not required perform any action, and your code should avoid generating exceptions or allowing exceptions to propagate to the calling method. Any exceptions that do reach the caller are ignored.

When you create a IfxRowsCopiedEventArgs delegate, you identify the method that will handle the event. To associate the event with your event handler, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see "Events and Delegates" in the .NET Framework SDK documentation.

**Reference**

"IfxRowsCopiedEventArgs Class" on page 5-563
"IfxBulkCopy.IfxRowsCopied Event" on page 5-38
"IfxBulkCopy Class" on page 5-19
"IBM.Data.Informix Namespace" on page 5-1

# IfxString Structure

Represents the CHAR, VARCHAR, LONG VARCHAR, GRAPHIC, VARGRAPHIC and LONG VARGRAPHIC Informix data types. Encapsulates the string .NET data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
Public Structure IfxString
[C#]
public struct IfxString
[C++]
public value class IfxString
```

## Example

[C#] The following example demonstrates how to retrieve character string column values from a table.

```
[C#]
  public static string getParam(IfxConnection conn)
  {
    string mySelectQuery = "SELECT * FROM EMPLOYEE WHERE WORKDEPT=(?)";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);
    IfxParameter p1 = new IfxParameter();
    IfxString dept = new IfxString("D11");
    p1.Value = dept;
    myCommand.Parameters.Add(p1);
    IfxDataReader reader = myCommand.ExecuteReader();

    if (reader.Read())
    {
      IfxString selectValue = reader.GetIfxString(3);

      if (!selectValue.IsNull) { return selectValue.ToString(); }
    }

    return "NULL";
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxString Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxString Members

Represents the CHAR, VARCHAR, LONG VARCHAR, GRAPHIC, VARGRAPHIC and LONG VARGRAPHIC Informix data types. Encapsulates the string .NET data type. The following tables list the members exposed by the IfxString structure.

### Public Fields

| Field | Description |
|---|---|
| Null | Null value for IfxString. |

### Public Constructors

| Constructor | Description |
|---|---|
| IfxString | Initializes a new IfxString structure with the specified value. |

### Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxString structure is null. |
| Value | Gets the value stored in the IfxString structure. |

### Public Methods

| Method | Description |
|---|---|
| ToString | Returns a string that represents the IfxString structure. |
| op_explicit | Converts the supplied IfxString structure to a string. |
| op_implicit | Converts the supplied string to IfxString. |

**Reference**

"IfxString Structure" on page 5-566
"IBM.Data.Informix Namespace" on page 5-1

## IfxString Fields

The fields of the IfxString class are listed here.

### Public Fields

| Field | Description |
|---|---|
| Null | Null value for IfxString. |

**Reference**

"IfxString Members" on page 5-567
"IfxString Structure" on page 5-566
"IBM.Data.Informix Namespace" on page 5-1

**IfxString.Null Field:**

Null value for IfxString.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxString
[C#]
public static readonly IfxString Null
[C++]
public:
static initonly IfxString Null
[JScript]
public static final var Null () : IfxString
```

**Remarks**

The value of this constant is NULL.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxString Structure" on page 5-566

"IBM.Data.Informix Namespace" on page 5-1

## IfxString Constructor

Initializes a new IfxString structure with the specified value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

### Syntax

```
[Visual Basic]
Public Sub New(value as String)
[C#]
public IfxString(string value);
[C++]
public: IfxString(string value);
[JScript]
public function IfxString(value : string);
```

### Parameters

**value** A character string to populate the IfxString instance.

### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxString Structure" on page 5-566

"IBM.Data.Informix Namespace" on page 5-1

## IfxString Methods

The methods of the IfxString class are listed here.

### Public Methods

| Method | Description |
|---|---|
|  ToString | Returns a string that represents the IfxString structure. |
|  op_explicit | Converts the supplied IfxString structure to a string. |
|  op_implicit | Converts the supplied string to IfxString. |

**Reference**

"IfxString Members" on page 5-567
"IfxString Structure" on page 5-566
"IBM.Data.Informix Namespace" on page 5-1

**IfxString.op_explicit Method:**

Converts the supplied IfxString structure to a character string.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Narrowing Operator CType (source As IfxString) As String
[C#]
public static implicit operator string (IfxString source)
[C++]
public:
static implicit operator string (IfxString source)
```

**Parameters**

**source** A IfxString structure to be converted to a character string.

**Return value**

A character string value converted from a IfxString instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxString Structure" on page 5-566
"IBM.Data.Informix Namespace" on page 5-1

**IfxString.op_implicit Method:**

Converts the supplied character string to IfxString.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Widening Operator CType (source As String) As IfxString
[C#]
public static implicit operator IfxString (string source)
[C++]
public:
static implicit operator IfxString (string source)
```

**Parameters**

**source** A character string value to be converted to IfxString.

**Return value**

A IfxString structure with the value of **source**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxString Structure" on page 5-566

"IBM.Data.Informix Namespace" on page 5-1

**IfxString.ToString Method:**

Returns a string that represents the IfxString structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxString structure.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxString Structure" on page 5-566

"IBM.Data.Informix Namespace" on page 5-1

## IfxString Properties

The properties of the IfxString class are listed here.

### Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxString structure is null. |
| Value | Gets the value stored in the IfxString structure. |

**Reference**

"IfxString Members" on page 5-567

"IfxString Structure" on page 5-566

"IBM.Data.Informix Namespace" on page 5-1

**IfxString.IsNull Property:**

Gets a value that indicates if the value stored in the IfxString structure is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxString Structure" on page 5-566

"IBM.Data.Informix Namespace" on page 5-1

**IfxString.Value Property:**

Gets the value stored in the IfxString structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Value As String
[C#]
public string Value {get;}
[C++]
public: __property string get_Value();
[JScript]
public function get Value() : string;
```

**Property value**

A character string representing the IfxString instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxString Structure" on page 5-566

"IBM.Data.Informix Namespace" on page 5-1

# IfxTime Structure

Represents the TIME Informix data type. Encapsulates the TimeSpan .NET data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
Public Structure IfxTime
[C#]
public struct IfxTime
[C++]
public value class IfxTime
```

## Example

[C#] The following example demonstrates how to retrieve a single TIME column value from a table.

```
[C#]
  public static string getParam(IfxConnection conn)
  {
    string mySelectQuery = "SELECT * FROM CL_SCHED";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);
    IfxDataReader reader = myCommand.ExecuteReader();
```

```
    if (reader.Read())
    {
      IfxTime selectValue = reader.GetIfxTime(3);

      if (!selectValue.IsNull) { return selectValue.ToString(); }
    }

    return "NULL";
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxTime Members

Represents the TIME Informix data type. Encapsulates the TimeSpan .NET data type. The following tables list the members exposed by the IfxTime structure.

### Public Fields

| Field | Description |
|---|---|
| Null | Null value for IfxTime. |
| MaxValue | Maximum value for IfxTime: 24:00:00. |
| MinValue | Minimum value for IfxTime: 00:00:00. |

### Public Constructors

| Constructor | Description |
|---|---|
| IfxTime() | Initializes a new IfxTime structure. |
| IfxTime(TimeSpan) | Initializes a new IfxTime structure with the specified TimeSpan value. |

### Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxTime structure is null. |

| Property | Description |
|---|---|
|  Value | Gets the value stored in the IfxTime structure in the form of a TimeStamp. |

### Public Methods

| Method | Description |
|---|---|
|  Parse | Converts the supplied String to IfxTime. |
|  ToString | Returns a string that represents the IfxTime structure. |
|  op_explicit | Converts the supplied IfxTime structure to a TimeStamp. |
|  op_implicit | Converts the supplied TimeStamp to IfxTime. |

**Reference**

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

## IfxTime Fields

The fields of the IfxTime structure are listed here.

### Public Fields

| Field | Description |
|---|---|
|  Null | Null value for IfxTime. |
|  MaxValue | Maximum value for IfxTime: 24:00:00. |
|  MinValue | Minimum value for IfxTime: 00:00:00. |

**Reference**

"IfxTime Members" on page 5-574

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

**IfxTime.MaxValue Field:**

Maximum value for IfxTime: 24:00:00.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MaxValue As IfxTime
[C#]
```

```
public static readonly IfxTime MaxValue
[C++]
public:
static initonly IfxTime MaxValue
[JScript]
public static final var MaxValue () : IfxTime
```

**Remarks**

The value of this constant is 24:00:00.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

**IfxTime.MinValue Field:**

Minimum value for IfxTime: 00:00:00.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MinValue As IfxTime
[C#]
public static readonly IfxTime MinValue
[C++]
public:
static initonly IfxTime MinValue
[JScript]
public static final var MinValue () : IfxTime
```

**Remarks**

The value of this constant is: 00:00:00.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

**IfxTime.Null Field:**

Null value for IfxTime.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxTime
[C#]
public static readonly IfxTime Null
[C++]
public:
static initonly IfxTime Null
[JScript]
public static final var Null () : IfxTime
```

**Remarks**

The value of this constant is `NULL`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Structure" on page 5-573
"IBM.Data.Informix Namespace" on page 5-1

## IfxTime Constructors

Initializes a new instance of the IfxTime class.

### Overload List

| Method | Description |
|---|---|
|  IfxTime() | Initializes a new IfxTime object. |
|  IfxTime(TimeSpan) | Initializes a new IfxTime object with the specified TimeSpan value. |

**Reference**

"IfxTime Members" on page 5-574
"IfxTime Structure" on page 5-573
"IBM.Data.Informix Namespace" on page 5-1

**IfxTime.IfxTime() Constructor:**

Initializes a new IfxTime object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New()
[C#]
public IfxTime();
```

```
[C++]
public: IfxTime();
[JScript]
public function IfxTime();
```

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

**IfxTime.IfxTime(TimeSpan) Constructor:**

Initializes a new IfxTime object with the specified TimeSpan value.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New(value as TimeSpan)
[C#]
public IfxTime(TimeSpan value);
[C++]
public: IfxTime(TimeSpan value);
[JScript]
public function IfxTime(value : TimeSpan);
```

**Parameters**

**Version information**

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

## IfxTime.IfxTime(string) Constructor

Initializes a new IfxTime structure with the specified value.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
Public Sub New(value as TimeSpan)
[C#]
public IfxTime(TimeSpan value);
[C++]
```

```
public: IfxTime(TimeSpan value);
[JScript]
public function IfxTime(value : TimeSpan);
```

### Parameters

**value** A System.Timestamp value to populate the IfxTime instance.

### Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

## IfxTime Methods

The methods of the IfxTime structure are listed here.

### Public Methods

| Method | Description |
|---|---|
|  Parse | Converts the supplied String to IfxTime. |
|  ToString | Returns a string that represents the IfxTime structure. |
|  op_explicit | Converts the supplied IfxTime structure to a TimeStamp. |
|  op_implicit | Converts the supplied TimeStamp to IfxTime. |

**Reference**

"IfxTime Members" on page 5-574

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

**IfxReal.op_explicit Method:**

Converts the supplied IfxReal structure to a TimeStamp.

**Namespace:**

IBM.Data.Informix

**Assembly:**

IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Narrowing Operator CType (source As IfxReal) As TimeStamp
[C#]
public static implicit operator TimeStamp (IfxReal source)
[C++]
public:
static implicit operator TimeStamp (IfxReal source)
```

**Parameters**

**source** A IfxReal structure to be converted to a TimeStamp.

**Return value**

A TimeStamp value converted from a IfxReal instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxReal Structure" on page 5-537

"IBM.Data.Informix Namespace" on page 5-1

**IfxTime.op_implicit Method:**

Converts the supplied TimeStamp to IfxTime.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Widening Operator CType (source As TimeStamp) As IfxTime
[C#]
public static implicit operator IfxTime (TimeStamp source)
[C++]
public:
static implicit operator IfxTime (TimeStamp source)
```

**Parameters**

**source** A TimeStamp to be converted to IfxTime.

**Return value**

A IfxTime structure with the value of **source**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

**IfxTime.Parse Method:**

Converts the supplied String to IfxTime.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Function Parse (szTime As String) As IfxTime
[C#]
public static IfxTime Parse (string szTime)
[C++]
public:
static IfxTime Parse (string szTime)
[JScript]
public static function Parse (szTime String ) : IfxTime
```

**Parameters**

**szTime** A String value to be converted to IfxTime. The string representation of the time must reflect the time format that corresponds to the territory code of the client application.

**Return value**

A IfxTime structure with the numeric value of **szTime**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

**IfxTime.ToString Method:**

Returns a string that represents the IfxTime structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxTime structure. The string representation of the time will reflect the time format that corresponds to the territory code of the client application.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

## IfxTime Properties

The properties of the IfxTime structure are listed here.

### Public Properties

| Property | Description |
|---|---|
|  IsNull | Gets a value that indicates if the value stored in the IfxTime structure is null. |
|  Value | Gets the value stored in the IfxTime structure in the form of a TimeStamp. |

**Reference**

"IfxTime Members" on page 5-574

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

**IfxTime.IsNull Property:**

Gets a value that indicates if the value stored in the IfxTime structure is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

### IfxTime.TimeSpanValue Property:

Gets the value stored in the IfxTime structure in the form of a TimeSpan structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

#### Syntax

```
[Visual Basic]
Public ReadOnly Property Value As TimeStamp
[C#]
public TimeStamp Value {get;}
[C++]
public: __property TimeStamp get_Value();
[JScript]
public function get Value() : TimeStamp;
```

#### Property value

A TimeStamp representing the IfxTime instance.

#### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

### IfxTime.Value Property:

Gets the value stored in the IfxTime structure in the form of a TimeStamp.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

#### Syntax

```
[Visual Basic]
Public ReadOnly Property Value As TimeStamp
[C#]
public TimeStamp Value {get;}
[C++]
public: __property TimeStamp get_Value();
[JScript]
public function get Value() : TimeStamp;
```

#### Property value

A TimeStamp representing the IfxTime instance.

#### Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTime Structure" on page 5-573

"IBM.Data.Informix Namespace" on page 5-1

# IfxDateTimeStructure

Represents the Informix DateTime data type. Encapsulates the DateTime .NET data type.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
Public Structure IfxDateTime
[C#]
public struct IfxDateTime
[C++]
public value class IfxDateTime
```

## Example

[C#] The following example demonstrates how to retrieve a single DateTime column value from a table.

```
[C#]
  public static string getParam(IfxConnection conn)
  {
    string mySelectQuery = "SELECT * FROM IN_TRAY";
    IfxCommand myCommand = new IfxCommand(mySelectQuery, conn);
    IfxDataReader reader = myCommand.ExecuteReader();

    if (reader.Read())
    {
      IfxDateTime selectValue = reader.GetIfxDateTime(0);

      if (!selectValue.IsNull) { return selectValue.ToString(); }
    }

    return "NULL";
  }
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDateTime Members" on page 5-585

"IBM.Data.Informix Namespace" on page 5-1

## IfxDateTime Members

Represents the Informix DateTime data type. Encapsulates the DateTime .NET data type. The following tables list the members exposed by the IfxDateTime structure.

### Public Fields

| Field | Description |
|---|---|
| Null | Null value for IfxDateTime. |
| MaxValue | Maximum value for IfxDateTime: 24:00:00.00000000000, December 31, 9999. |
| MinValue | Minimum value for IfxDateTime: 00:00:00.000000000000, January 1, 0001. |

### Public Constructors

| Constructor | Description |
|---|---|
| IfxDateTime() | Initializes a new IfxDateTime set to `null`. |
| IfxDateTime(int year, int month, int day, int hour, int minute, int second, int microsecond) | Initializes a new IfxDateTime structure with the specified integer values. |
| IfxDateTime(DateTime value) | Initializes a new IfxDateTime structure with the specified DateTime value. |

### Public Properties

| Property | Description |
|---|---|
| IsNull | Gets a value that indicates if the value stored in the IfxDateTime object is null. |
| Value | Gets the value stored in the IfxDateTime object in the form of a System.DateTime object. |

### Public Methods

| Method | Description |
|---|---|
| Parse | Converts the supplied String to IfxDateTime. |
| ToString | Returns a string that represents the IfxDateTime object. |
| op_explicit | Converts the supplied IfxDateTime structure to a DateTime. |
| op_implicit | Converts the supplied DateTime to IfxDateTime. |

**Reference**

"IfxDateTimeStructure" on page 5-584

"IBM.Data.Informix Namespace" on page 5-1

## IfxDateTimeFields

The fields of the IfxDateTime structure are listed here.

### Public Fields

| Field | Description |
|---|---|
| Null | Null value for IfxDateTime. |
| MaxValue | Maximum value for IfxDateTime: 24:00:00.00000000000, December 31, 9999. |
| MinValue | Minimum value for IfxDateTime: 00:00:00.000000000000, January 1, 0001. |

**Reference**

"IfxDateTime Members" on page 5-585
"IfxDateTimeStructure" on page 5-584
"IBM.Data.Informix Namespace" on page 5-1

**IfxDateTime.MaxValue Field:**

Maximum value for IfxDateTime: 24:00:00.00000000000, December 31, 9999.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MaxValue As IfxDateTime
[C#]
public static readonly IfxDateTime MaxValue
[C++]
public:
static initonly IfxDateTime MaxValue
[JScript]
public static final var MaxValue () : IfxDateTime
```

**Remarks**

The value of this constant is 24:00:00.00000000000, December 31, 9999.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDateTimeStructure" on page 5-584
"IBM.Data.Informix Namespace" on page 5-1

**IfxDateTime.MinValue Field:**

Minimum value for IfxDateTime: 00:00:00.000000000000, January 1, 0001.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly MinValue As IfxDateTime
[C#]
public static readonly IfxDateTime MinValue
[C++]
public:
static initonly IfxDateTime MinValue
[JScript]
public static final var MinValue () : IfxDateTime
```

**Remarks**

The value of this constant is: 00:00:00.000000000000, January 1, 0001.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDateTimeStructure" on page 5-584
"IBM.Data.Informix Namespace" on page 5-1

**IfxDateTime.Null Field:**

Null value for IfxDateTime.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared ReadOnly Null As IfxDateTime
[C#]
public static readonly IfxDateTime Null
[C++]
public:
static initonly IfxDateTime Null
[JScript]
public static final var Null () : IfxDateTime
```

**Remarks**

The value of this constant is `NULL`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDateTimeStructure" on page 5-584

"IBM.Data.Informix Namespace" on page 5-1

## IfxDateTimeConstructors

Initializes a new instance of the IfxDateTime class.

### Public Constructors

| Constructor | Description |
|---|---|
| IfxDateTime() | Initializes a new IfxDateTime set to `null`. |
| IfxDateTime(int year, int month, int day, int hour, int minute, int second, int microsecond) | Initializes a new IfxDateTime structure with the specified integer values. |
| IfxDateTime(DateTime value) | Initializes a new IfxDateTime structure with the specified DateTime value. |

**Reference**

"IfxDateTime Members" on page 5-585
"IfxDateTimeStructure" on page 5-584
"IBM.Data.Informix Namespace" on page 5-1

**IfxDateTime.IfxDateTime(int year, int month, int day, int hour, int minute, int second, int microsecond) Constructor:**

Initializes a new IfxDateTime structure with the specified integer values.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New(year as Integer, month as Integer, _
    day as Integer, hour as Integer, _
    minute as Integer, second as Integer, _
    microsecond as Integer _
)
[C#]
public IfxDateTime(int year, int month,
    int day, int hour,
    int minute, int second,
    int microsecond
);
[C++]
public: IfxDateTime(int year,int month,
    int day, int hour,
    int minute, int second,
    int microsecond
);
[JScript]
public function IfxDateTime(year : int, month : int,
    year : int, month : int,
    day : int, hour : int,
    minute : int, second : int,
    microsecond : int
 );
```

**Parameters**

**year** An integer value to provide the year for the IfxDateTime instance.

**month** An integer value to provide the month for the IfxDateTime instance.

**day** An integer value to provide the day for the IfxDateTime instance.

**hour** An integer value to provide the hour for the IfxDateTime instance.

**minute** An integer value to provide the minute for the IfxDateTime instance.

**second** An integer value to provide the second for the IfxDateTime instance.

**microsecond**
An integer value to provide the microsecond for the IfxDateTime instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDateTimeStructure" on page 5-584

"IBM.Data.Informix Namespace" on page 5-1

**IfxDateTime.IfxDateTime(DateTime value) Constructor:**

Initializes a new IfxDateTime structure with the specified DateTime value.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Sub New(dtvalue as DateTime)
[C#]
public IfxDateTime(DateTime dtvalue);
[C++]
public: IfxDateTime(DateTime dtvalue);
[JScript]
public function IfxDateTime(dtvalue : DateTime);
```

**Parameters**

**value** A DateTime value to populate the IfxDateTime instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDateTimeStructure" on page 5-584

"IBM.Data.Informix Namespace" on page 5-1

## IfxDateTimeMethods

The methods of the IfxDateTime class are listed here.

## Public Methods

| Method | Description |
|---|---|
|  Parse | Converts the supplied String to IfxDateTime. |
|  ToString | Returns a string that represents the IfxDateTime object. |
|  op_explicit | Converts the supplied IfxDateTime structure to a DateTime. |
|  op_implicit | Converts the supplied DateTime to IfxDateTime. |

**Reference**

"IfxDateTime Members" on page 5-585

"IfxDateTimeStructure" on page 5-584

"IBM.Data.Informix Namespace" on page 5-1

### IfxDateTime.op_explicit Method:

Converts the supplied IfxDateTime structure to a DateTime.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Narrowing Operator CType (source As IfxDateTime) As DateTime
[C#]
public static implicit operator DateTime (IfxDateTime source)
[C++]
public:
static implicit operator DateTime (IfxDateTime source)
```

**Parameters**

**source** A IfxDateTime structure to be converted to a DateTime.

**Return value**

A DateTime value converted from a IfxDateTime instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDateTimeStructure" on page 5-584

"IBM.Data.Informix Namespace" on page 5-1

### IfxDateTime.op_implicit Method:

Converts the supplied DateTime to IfxDateTime.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Widening Operator CType (source As Short) As IfxDateTime
[C#]
public static implicit operator IfxDateTime (short source)
[C++]
public:
static implicit operator IfxDateTime (short source)
```

**Parameters**

**source** A DateTime value to be converted to IfxDateTime.

**Return value**

A IfxDateTime structure with the value of **source**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxInt16 Structure" on page 5-428

"IBM.Data.Informix Namespace" on page 5-1

**IfxDateTime.Parse Method:**

Converts the supplied String to IfxDateTime.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Shared Function Parse (szTimeStamp As String) As IfxDateTime
[C#]
public static IfxDateTime Parse (string szTimeStamp)
[C++]
public:
static IfxDateTime Parse (string szTimeStamp)
[JScript]
public static function Parse (szTimeStamp String ) : IfxDateTime
```

**Parameters**

**szTimeStamp**
A String value to be converted to IfxDateTime. The string representation of the timestamp must reflect the date format that corresponds to the territory code of the client application or be in the IfxDateTime format, *yyyy-mm-dd-hh.mm.ss.ffffff*.

**Return value**

A IfxDateTime structure with the numeric value of **szTimeStamp**.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDateTimeStructure" on page 5-584

"IBM.Data.Informix Namespace" on page 5-1

**IfxDateTime.ToString Method:**

Returns a string that represents the IfxDateTime structure.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public Overrides Function ToString As String
[C#]
public override string ToString ()
[C++]
public:
virtual String^ ToString () override
[JScript]
public override function ToString () : String
```

**Return value**

A string representing the value of the IfxDateTime structure. The string representation of the timestamp will reflect the timestamp format that corresponds to the territory code of the client application.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDateTimeStructure" on page 5-584

"IBM.Data.Informix Namespace" on page 5-1

## IfxDateTimeProperties

The properties of the IfxDateTime structure are listed here.

### Public Properties

| Property | Description |
|---|---|
|  IsNull | Gets a value that indicates if the value stored in the IfxDateTime object is null. |

| Property | Description |
|---|---|
|  Value | Gets the value stored in the IfxDateTime object in the form of a System.DateTime object. |

**Reference**

"IfxDateTime Members" on page 5-585

"IfxDateTimeStructure" on page 5-584

"IBM.Data.Informix Namespace" on page 5-1

**IfxDateTime.IsNull Property:**

Gets a value that indicates if the value stored in the IfxDateTime structure is null.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsNull As Boolean
[C#]
public bool IsNull {get;}
[C++]
public: __property bool get_IsNull();
[JScript]
public final function get IsNull() : boolean
```

**Property value**

`true` if Value is null; otherwise, `false`.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDateTimeStructure" on page 5-584

"IBM.Data.Informix Namespace" on page 5-1

**IfxDateTime.Value Property:**

Gets the value stored in the IfxDateTime object in the form of a System.DateTime object.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Value As TimeStamp
[C#]
public TimeStamp Value {get;}
```

```
[C++]
public: __property TimeStamp get_Value();
[JScript]
public function get Value() : TimeStamp;
```

**Property value**

A TimeStamp representing the IfxDateTime instance.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxDateTimeStructure" on page 5-584

"IBM.Data.Informix Namespace" on page 5-1

# IfxTransaction Class

Represents an SQL transaction to be made at a database. This class cannot be inherited.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## .NET Framework 2.0 3.0, 3.5, and 4.0 Inheritance hierarchy

System.Object
  System.MarshalByRefObject
    System.Data.Common.DbTransaction
      IBM.Data.Informix.IfxTransaction

## .NET Framework 2.0 3.0, 3.5, and 4.0 Syntax

```
[Visual Basic]
NotInheritable Public Class IfxTransaction
   Inherits DbTransaction
   Implements IDbTransaction, IDisposable
[C#]
public sealed class IfxTransaction : DbTransaction,
   IDbTransaction, IDisposable
[C++]
public __gc __sealed class IfxTransaction : public
   DbTransaction, IDbTransaction, IDisposable
[JScript]
public class IfxTransaction extends DbTransaction implements
   IDbTransaction, IDisposable
```

## Remarks

The application creates an IfxTransaction object by calling BeginTransaction on the IfxConnection object. All subsequent operations associated with the transaction (for example, committing or aborting the transaction), are performed on the IfxTransaction object.

## Example

[Visual Basic, C#] The following example creates a IfxConnection and a IfxTransaction. It also demonstrates how to use the IfxConnection.BeginTransaction, IfxTransaction.Commit, and IfxTransaction.Rollback methods.

```
[Visual Basic]
Public Sub RunIfxTransaction(myConnString As String)
    Dim myConnection As New IfxConnection(myConnString)
    myConnection.Open()

    Dim myCommand As New IfxCommand()
    Dim myTrans As IfxTransaction

    ' Start a local transaction
    myTrans = myConnection.BeginTransaction(IsolationLevel.ReadCommitted)
    ' Assign transaction object for a pending local transaction
    myCommand.Transaction = myTrans

    Try
        myCommand.CommandText = "Insert into org(
          DEPTNUMB, DEPTNAME, MANAGER, DIVISION,LOCATION)
          VALUES (100, 'Head Office', 160, 'Corporate', 'New York')"
        myCommand.ExecuteNonQuery()
        myCommand.CommandText = "Insert into org(
          DEPTNUMB, DEPTNAME, MANAGER, DIVISION,LOCATION)
          VALUES (101, 'New England', 50, 'Eastern', 'Boston')"
        myCommand.ExecuteNonQuery()
        myTrans.Commit()
        Console.WriteLine("Both records are written to database.")
    Catch e As Exception
        myTrans.Rollback()
        Console.WriteLine(e.ToString())
        Console.WriteLine("Neither record was written to database.")
    Finally
        myConnection.Close()
    End Try
End Sub

[C#]
public void RunIfxTransaction(string myConnString)
{
   IfxConnection myConnection = new IfxConnection(myConnString);
   myConnection.Open();

   IfxCommand myCommand = new IfxCommand();
   IfxTransaction myTrans;

   // Start a local transaction
   myTrans = myConnection.BeginTransaction(IsolationLevel.ReadCommitted);
   // Assign transaction object for a pending local transaction
   myCommand.Transaction = myTrans;

   try
   {
     myCommand.CommandText = "Insert into org(
       DEPTNUMB, DEPTNAME, MANAGER, DIVISION,LOCATION)
       VALUES (100, 'Head Office', 160, 'Corporate', 'New York')";
     myCommand.ExecuteNonQuery();
     myCommand.CommandText = "Insert into org(
       DEPTNUMB, DEPTNAME, MANAGER, DIVISION,LOCATION)
       VALUES (101, 'New England', 50, 'Eastern', 'Boston')";
     myCommand.ExecuteNonQuery();
     myTrans.Commit();
     Console.WriteLine("Both records are written to database.");
   }
   catch(Exception e)
```

```
    {
      myTrans.Rollback();
      Console.WriteLine(e.ToString());
      Console.WriteLine("Neither record was written to database.");
    }
    finally
    {
      myConnection.Close();
    }
}
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTransaction Members"

"IBM.Data.Informix Namespace" on page 5-1

"IfxDataAdapter Class" on page 5-253

"IfxConnection Class" on page 5-157

## IfxTransaction Members

IfxTransaction overview

### Public Properties

| Property | Description |
|---|---|
| DbConnection | Specifies the DbConnection object associated with the transaction. |
| Connection | Specifies the IfxConnection object associated with the transaction. |
| IsolationLevel | Specifies the IsolationLevel for this transaction. |

### Public Methods

| Method | Description |
|---|---|
| Commit | Commits the database transaction. |
| CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |

| Method | Description |
|---|---|
| GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| Rollback | Rolls back a transaction from a pending state. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

### Protected Methods

| Method | Description |
|---|---|
| Dispose | Releases the unmanaged resources used by the IfxTransaction. |
| Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxTransaction Class" on page 5-594

"IBM.Data.Informix Namespace" on page 5-1

## IfxTransaction Methods

The methods of the IfxTransaction class are listed here. For a complete list of IfxTransaction class members, see the IfxTransaction Members topic.

### Public Methods

| Method | Description |
|---|---|
| Commit | Commits the database transaction. |
| CreateObjRef (inherited from MarshalByRefObject) | Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. |
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |

| Method | Description |
|---|---|
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetLifetimeService (inherited from MarshalByRefObject) | Retrieves the current lifetime service object that controls the lifetime policy for this instance. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| InitializeLifetimeService (inherited from MarshalByRefObject) | Obtains a lifetime service object to control the lifetime policy for this instance. |
| Rollback | Rolls back a transaction from a pending state. |
| ToString (inherited from Object) | Returns a String that represents the current Object. |

## Protected Methods

| Method | Description |
|---|---|
| Dispose | Releases the unmanaged resources used by the IfxTransaction. |
| Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxTransaction Class" on page 5-594

"IBM.Data.Informix Namespace" on page 5-1

**IfxTransaction.Commit Method:**

Commits the database transaction.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Public Sub Commit()
[C#]
public void Commit();
```

```
[C++]
public: __sealed void Commit();
[JScript]
public function Commit();
```

### Exceptions

| Exception type | Condition |
|---|---|
| Exception | An error occurred while trying to commit the transaction. |
| InvalidOperationException | The transaction has already been committed or rolled back.<br><br>-or-<br><br>The connection is broken. |

### Example

[Visual Basic, C#] The following example creates a IfxConnection and a IfxTransaction. It also demonstrates how to use the IfxConnection.BeginTransaction, Commit, and IfxTransaction.Rollback methods.

```
[Visual Basic]
Public Sub RunIfxTransaction(myConnString As String)
    Dim myConnection As New IfxConnection(myConnString)
    myConnection.Open()

    Dim myCommand As New IfxCommand()
    Dim myTrans As IfxTransaction

    ' Start a local transaction
    myTrans = myConnection.BeginTransaction()
    ' Assign transaction object for a pending local transaction
    myCommand.Transaction = myTrans

    Try
        myCommand.CommandText = "Insert into org(
          DEPTNUMB, DEPTNAME, MANAGER, DIVISION,LOCATION)
          VALUES (100, 'Head Office', 160, 'Corporate', 'New York')"
        myCommand.ExecuteNonQuery()
        myCommand.CommandText = "Insert into org(
          DEPTNUMB, DEPTNAME, MANAGER, DIVISION,LOCATION)
          VALUES (101, 'New England', 50, 'Eastern', 'Boston')"
        myCommand.ExecuteNonQuery()
        myTrans.Commit()
        Console.WriteLine("Both records are written to database.")
    Catch e As Exception
        myTrans.Rollback()
        Console.WriteLine(e.ToString())
        Console.WriteLine("Neither record was written to database.")
    Finally
        myConnection.Close()
    End Try
End Sub

[C#]
public void RunIfxTransaction(string myConnString)
{
   IfxConnection myConnection = new IfxConnection(myConnString);
   myConnection.Open();

   IfxCommand myCommand = new IfxCommand();
   IfxTransaction myTrans;
```

```
   // Start a local transaction
   myTrans = myConnection.BeginTransaction();
   // Assign transaction object for a pending local transaction
   myCommand.Transaction = myTrans;

   try
   {
     myCommand.CommandText = "Insert into org(
       DEPTNUMB, DEPTNAME, MANAGER, DIVISION,LOCATION)
       VALUES (100, 'Head Office', 160, 'Corporate', 'New York')";
     myCommand.ExecuteNonQuery();
     myCommand.CommandText = "Insert into org(
       DEPTNUMB, DEPTNAME, MANAGER, DIVISION,LOCATION)
       VALUES (101, 'New England', 50, 'Eastern', 'Boston')";
     myCommand.ExecuteNonQuery();
     myTrans.Commit();
     Console.WriteLine("Both records are written to database.");
   }
   catch(Exception e)
   {
     myTrans.Rollback();
     Console.WriteLine(e.ToString());
     Console.WriteLine("Neither record was written to database.");
   }
   finally
   {
     myConnection.Close();
   }
}
```

**Reference**

"IfxTransaction Class" on page 5-594

"IfxTransaction Members" on page 5-596

"IBM.Data.Informix Namespace" on page 5-1

"IfxConnection.BeginTransaction Method" on page 5-170

"IfxTransaction.Rollback Method" on page 5-601

### IfxTransaction.Dispose Method:

Releases the unmanaged resources used by the DbTransaction.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Overrides Overloads Protected Sub Dispose( _
   ByVal disposing As Boolean _
)
[C#]
protected override void Dispose(
   bool disposing
);
[C++]
protected: void Dispose(
   bool disposing
);
```

```
[JScript]
protected override function Dispose(
   disposing : Boolean
);
```

**Parameters**

**disposing**
`true` to release both managed and unmanaged resources; `false` to release only unmanaged resources.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTransaction Class" on page 5-594

"IBM.Data.Informix Namespace" on page 5-1

**IfxTransaction.Rollback Method:**

Rolls back a transaction from a pending state.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
NotOverridable Public Sub Rollback() Implements _
   IDbTransaction.Rollback
[C#]
public void Rollback();
[C++]
public: __sealed void Rollback();
[JScript]
public function Rollback();
```

**Implements:**
IDbTransaction.Rollback

**Exceptions**

| Exception type | Condition |
|---|---|
| Exception | An error occurred while trying to commit the transaction. |
| InvalidOperationException | The transaction has already been committed or rolled back.<br><br>-or-<br><br>The connection is broken. |

**Remarks**

The transaction can be rolled back only from a pending state (after IfxConnection.BeginTransaction has been called, but before IfxTransaction.Commit is called).

**Example**

[Visual Basic, C#] The following example creates a IfxConnection and a IfxTransaction. It also demonstrates how to use the IfxConnection.BeginTransaction, IfxTransaction.Commit, and Rollback methods.

```
[Visual Basic]
Public Sub RunIfxTransaction(myConnString As String)
    Dim myConnection As New IfxConnection(myConnString)
    myConnection.Open()

    Dim myCommand As New IfxCommand()
    Dim myTrans As IfxTransaction

    ' Start a local transaction
    myTrans = myConnection.BeginTransaction()
    ' Assign transaction object for a pending local transaction
    myCommand.Transaction = myTrans

    Try
        myCommand.CommandText = "Insert into org(
          DEPTNUMB, DEPTNAME, MANAGER, DIVISION,LOCATION)
          VALUES (100, 'Head Office', 160, 'Corporate', 'New York')"
        myCommand.ExecuteNonQuery()
        myCommand.CommandText = "Insert into org(
          DEPTNUMB, DEPTNAME, MANAGER, DIVISION,LOCATION)
          VALUES (101, 'New England', 50, 'Eastern', 'Boston')"
        myCommand.ExecuteNonQuery()
        myTrans.Commit()
        Console.WriteLine("Both records are written to database.")
    Catch e As Exception
        myTrans.Rollback()
        Console.WriteLine(e.ToString())
        Console.WriteLine("Neither record was written to database.")
    Finally
        myConnection.Close()
    End Try
End Sub

[C#]
public void RunIfxTransaction(string myConnString)
{
   IfxConnection myConnection = new IfxConnection(myConnString);
   myConnection.Open();

   IfxCommand myCommand = new IfxCommand();
   IfxTransaction myTrans;

   // Start a local transaction
   myTrans = myConnection.BeginTransaction();
   // Assign transaction object for a pending local transaction
   myCommand.Transaction = myTrans;

   try
   {
     myCommand.CommandText = "Insert into org(
      DEPTNUMB, DEPTNAME, MANAGER, DIVISION,LOCATION)
      VALUES (100, 'Head Office', 160, 'Corporate', 'New York')";
     myCommand.ExecuteNonQuery();
     myCommand.CommandText = "Insert into org(
```

```
      DEPTNUMB, DEPTNAME, MANAGER, DIVISION,LOCATION)
      VALUES (101, 'New England', 50, 'Eastern', 'Boston')";
    myCommand.ExecuteNonQuery();
    myTrans.Commit();
    Console.WriteLine("Both records are written to database.");
  }
  catch(Exception e)
  {
    myTrans.Rollback();
    Console.WriteLine(e.ToString());
    Console.WriteLine("Neither record was written to database.");
  }
  finally
  {
    myConnection.Close();
  }
}
```

**Reference**

"IfxTransaction Class" on page 5-594

"IfxTransaction Members" on page 5-596

"IBM.Data.Informix Namespace" on page 5-1

"IfxTransaction.Commit Method" on page 5-598

"IfxConnection.BeginTransaction Method" on page 5-170

## IfxTransaction Properties

The properties of the IfxTransaction class are listed here. For a complete list of IfxTransaction class members, see the IfxTransaction Members topic.

### Public Properties

| Property | Description |
|---|---|
| DbConnection | Specifies the DbConnection object associated with the transaction. |
| Connection | Specifies the IfxConnection object associated with the transaction. |
| IsolationLevel | Specifies the IsolationLevel for this transaction. |

**Reference**

"IfxTransaction Class" on page 5-594

"IBM.Data.Informix Namespace" on page 5-1

**IfxTransaction.Connection Property:**

Specifies the IfxConnection object associated with the transaction.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property Connection As IfxConnection

[C#]
```

```
public IfxConnection
 Connection {get;}
[C++]
public: __property IfxConnection
* get_Connection();
[JScript]
public function get Connection() : IfxConnection
;
```

**Property value**

The IfxConnection object associated with the transaction.

**Remarks**

A single application may have multiple database connections, each with its own transaction. This property enables you to determine the connection object associated with a particular transaction.

**Reference**

"IfxTransaction Class" on page 5-594

"IfxTransaction Members" on page 5-596

"IBM.Data.Informix Namespace" on page 5-1

**IfxTransaction.DbConnection Property:**

Specifies the DbConnection object associated with the transaction.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property DbConnection As DbConnection
[C#]
public DbConnection DbConnection {get;}
[C++]
public: __property DbConnection* get_DbConnection();
[JScript]
public function get DbConnection() : DbConnection;
```

**Property value**

The DbConnection object associated with the transaction.

**Remarks**

A single application may have multiple database connections, each with its own transaction. This property enables you to determine the connection object associated with a particular transaction.

**Version information**

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTransaction Class" on page 5-594
"IBM.Data.Informix Namespace" on page 5-1

**IfxTransaction.IsolationLevel Property:**

Specifies the IsolationLevel for this transaction.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

**Syntax**

```
[Visual Basic]
Public ReadOnly Property IsolationLevel As IsolationLevel  _
   Implements IDbTransaction.IsolationLevel
[C#]
public IsolationLevel IsolationLevel {get;}
[C++]
public: __property IsolationLevel get_IsolationLevel();
[JScript]
public function get IsolationLevel() : IsolationLevel;
```

**Implements:**
IDbTransaction.IsolationLevel

**Property value**

The IsolationLevel for this transaction. The default is ReadCommitted.

**Remarks**

Parallel transactions are not supported. Therefore, the IsolationLevel applies to the entire transaction.

Mappings between .NET Isolation enumerations and DB2 Isolation Levels are as follows:

| .NET Isolation Enumeration | DB2 Isolation Levels |
|---|---|
| ReadCommitted | Cursor Stability (default) |
| ReadUncommitted | Uncommitted Read |
| RepeatableRead | Read Stability |
| Serializable | Repeatable Read |
| Chaos | not supported |

**Reference**

"IfxTransaction Class" on page 5-594
"IfxTransaction Members" on page 5-596
"IBM.Data.Informix Namespace" on page 5-1

## IfxTruncateException Class

The exception that is generated when the Value property for a IfxDecimal or IfxDecimalFloat instance is accessed and the precision of this value exceeds that of a .NET Decimal variable.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
  System.Exception
    System.SystemException
      IBM.Data.Informix.IfxTypeException
        IBM.Data.Informix.IfxTruncateException

## Syntax

```
[Visual Basic]
Public Class IfxTruncateException
[C#]
public class IfxTruncateException
[C++]
public __gc class IfxTruncateException
[JScript]
public class IfxTruncateException
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTruncateException Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxTruncateException Members

The exception that is generated when the Value property for a IfxDecimal or IfxDecimalFloat instance is accessed and the precision of this value exceeds that of a .NET Decimal variable. The following tables list the members exposed by the IfxTruncateException class.

### Public Properties

| Property | Description |
|---|---|
| HelpLink (inherited from Exception) | Gets or sets a link to the help file associated with this exception. |
| InnerException (inherited from Exception) | Gets the Exception instance that caused the current exception. |
| Message (inherited from Exception | Gets the text describing the error. |
| Source (inherited from Exception | Gets the name of the application or the object that generated the error. |

| Property | Description |
|---|---|
| StackTrace (inherited from Exception) | Gets a string representation of the frames on the call stack at the time the current exception was thrown. |
| TargetSite (inherited from Exception) | Gets the method that throws the current exception. |

## Public Methods

| Method | Description |
|---|---|
| Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
| GetBaseException (inherited from Exception) | When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetObjectData (inherited from Object) | Overridden. See Exception.GetObjectData. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| ToString (inherited from Exception) | Overridden. Creates and returns a string representation of the current exception. |

## Protected Properties

| Property | Description |
|---|---|
| HResult (inherited from Exception) | Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. |

## Protected Methods

| Method | Description |
|---|---|
| Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxTruncateException Class" on page 5-605
"IBM.Data.Informix Namespace" on page 5-1

# IfxType Enumeration

Specifies the data type of a field, property, or IfxParameter.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Syntax

```
[Visual Basic]
<Serializable>
Public Enum IfxType
[C#]
[Serializable]
public enum IfxType
[C++]
[Serializable]
__value public enum IfxType
[JScript]
public
   Serializable
enum IfxType
```

## Remarks

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

| Category | IfxType Data Type | Informix Data Type | .NET Framework Data Type | IfxTypes Namespace Classes and Structures |
|---|---|---|---|---|
| Character data | Char | CHAR | String | IfxString |
| | VarChar | VARCHAR | String | IfxString |
| | LongVarChar | LVARCHAR | String | IfxString |
| Date/Time data | Date | DATETIME (date precision) | DateTime | IfxDate |
| | Time | DATETIME (time precision) | TimeSpan | IfxTime |
| | Datetime | DATETIME (date and time precision) | DateTime | IfxTimeStamp |
| LOB data | Blob | BLOB, BYTE | Byte[] | IfxBlob |
| | Clob | CLOB, TEXT | String | IfxClob |

1. Date and Time objects can be timestamp string literals. Timestamp objects can be date string literals. For information on timestamp string literal usage, see"IfxParameter Class" on page 5-457.

| Category | IfxType Data Type | Informix Data Type | .NET Framework Data Type | IfxTypes Namespace Classes and Structures |
|---|---|---|---|---|
| Numeric data | SmallInt | BOOLEAN, SMALLINT | Int16 | IfxInt16 |
| | Integer | INT, INTEGER, SERIAL | Int32 | IfxInt32 |
| 2 | BigInt, BigSerial | BIGINT, BIGSERIAL, INT8, SERIAL8 | Int64 | IfxInt64 |
| | Real | REAL, SMALLFLOAT | Single | IfxReal |
| | Double | DECIMAL (≤ 29), DOUBLE PRECISION | Double | IfxDouble |
| | Float | DECIMAL (32), FLOAT | Double | IfxDouble |
| | Decimal | DECIMAL | Decimal | IfxDecimal |
| | Money | MONEY | Decimal | IfxDecimal |
| | Numeric | DECIMAL (≤ 29), NUMERIC | Decimal | IfxDecimal |

**Note:** Starting in IBM DB2 Version 9.7 FixPack 3, a new member by name Cursor will be introduced in the DB2Type enumeration. This member should be used when binding an output parameter of the type cursor.

**Limitation**

The ParameterCollection.Value will always return a DataReader object, regardless of the type of cursor opened in the stored procedure.

Not all Informix data types are supported, the following IfxType enumeration values will throw an exception if they are used at run time:

- Char1
- Collection
- IntervalDayFraction
- IntervalMonth
- List
- MultiSet
- Row
- Set

For IfxSmartLOBCreateTimeFlags, IfxSmartLOBFileLocation, IfxSmartLOBWhence, IfxSmartLOBLockMode and IfxSmartLOBOpenMode, the methods and properties taking or returning those enumerations will throw not implemented exceptions.

## Version information

**.NET Framework version**

Supported in: 2.0, 3.0, 3.5, and 4.0

## Members

| Member | Description |
|---|---|
| BigInt | A 64-bit integer. Represents the BIGINT, BIGSERIAL, INT8, SERIAL8 Informix data types. |
| BigSerial | A 64-bit integer. Represents the BIGINT, BIGSERIAL, INT8, SERIAL8 Informix data types. |
| Binary | An array of bytes. Represents the CHAR, VARCHAR, and LVARCHAR Informix data types. |
| Blob | An array of bytes. Represents the BLOB and BYTE Informixdata types. |
| Char | A single character. Represents the CHAR Informix data type. |
| Clob | A large string of characters. Represents the CLOB and TEXT Informix data type. |
| Date | A string representing a date. Represents the DATETIME (date precision) Informix data type. |
| Decimal | A decimal value. Represents the MONEY Informix data type. |
| Double | A double-precision floating point value. Represents the DECIMAL (≤31), DOUBLE PRECISION Informix data type. |
| Float | A single-precision floating point value. Represents the DECIMAL (32), FLOAT Informix data type. |
| Integer | A 32-bit integer. Represents the INT, INTEGER, and SERIAL Informix data type. |
| Numeric | A decimal value. Represents the DECIMAL (≤31), NUMERIC Informix data type. |
| Real | A floating point number. Represents the REAL, SMALLFLOAT Informix data type. |
| SmallInt | A 16-bit integer. Represents the BOOLEAN, SMALLINT Informix data type. |
| Time | A string representing the time of day. Represents the DATETIME (time precision) Informix data type. |
| Timestamp | A string representing a timestamp. Represents the DATETIME (date and time precision) Informix data type. |
| VarChar | A string of characters. Represents the VARCHAR Informix data type. |
| Xml | An XML document. Represents the XML DB2 data type. |

**Reference**

"IBM.Data.Informix Namespace" on page 5-1

# IfxTypeException Class

The sub-classes IfxFormatException, IfxNullValueException, IfxTruncateException represent the exceptions that can be thrown during the incorrect use of certain IBM.Data.Informix structure or class instances.

**Namespace:**
IBM.Data.Informix

**Assembly:**
IBM.Data.Informix (in IBM.Data.Informix.dll)

## Inheritance hierarchy

System.Object
  System.Exception
    System.SystemException
      IBM.Data.Informix.IfxTypeException

## Syntax

```
[Visual Basic]
Public Class IfxTypeException
[C#]
public class IfxTypeException
[C++]
public __gc class IfxTypeException
[JScript]
public class IfxTypeException
```

## Thread safety

Any public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Any instance members are not guaranteed to be thread safe.

## Version information

**.NET Framework version**
Supported in: 2.0, 3.0, 3.5, and 4.0

**Reference**

"IfxTypeException Members"

"IBM.Data.Informix Namespace" on page 5-1

## IfxTypeException Members

The sub-classes IfxFormatException, IfxNullValueException, IfxTruncateException represent the exceptions that can be thrown during the incorrect use of certain IBM.Data.IfxTypes structure or class instances. The following tables list the members exposed by the IfxTypeException class.

### Public Properties

| Property | Description |
|---|---|
| HelpLink (inherited from Exception) | Gets or sets a link to the help file associated with this exception. |
| InnerException (inherited from Exception) | Gets the Exception instance that caused the current exception. |

| Property | Description |
|---|---|
|  Message (inherited from Exception | Gets the text describing the error. |
|  Source (inherited from Exception | Gets the name of the application or the object that generated the error. |
|  StackTrace (inherited from Exception) | Gets a string representation of the frames on the call stack at the time the current exception was thrown. |
|  TargetSite (inherited from Exception) | Gets the method that throws the current exception. |

## Public Methods

| Method | Description |
|---|---|
|  Equals (inherited from Object) | Overloaded. Determines whether two Object instances are equal. |
|  GetBaseException (inherited from Exception) | When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. |
|  GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
|  GetObjectData (inherited from Object) | Overridden. See Exception.GetObjectData. |
|  GetType (inherited from Object) | Gets the Type of the current instance. |
|  ToString (inherited from Exception) | Overridden. Creates and returns a string representation of the current exception. |

## Protected Properties

| Property | Description |
|---|---|
|  HResult (inherited from Exception) | Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. |

## Protected Methods

| Method | Description |
|---|---|
|  Finalize (inherited from Object) | Overridden. Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.<br><br>In C# and Microsoft(R) Visual C++(R), finalizers are expressed using destructor syntax. |
|  MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

**Reference**

"IfxTypeException Class" on page 5-611
"IBM.Data.Informix Namespace" on page 5-1

# Appendix. Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability.

## Accessibility features for IBM Informix products

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

### Accessibility features

The following list includes the major accessibility features in IBM Informix products. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers.
- The attachment of alternative input and output devices.

**Tip:** The information center and its related publications are accessibility-enabled for the IBM Home Page Reader. You can operate all features by using the keyboard instead of the mouse.

### Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

### Related accessibility information

IBM is committed to making our documentation accessible to persons with disabilities. Our publications are available in HTML format so that they can be accessed with assistive technology such as screen reader software.

You can view the publications in Adobe Portable Document Format (PDF) by using the Adobe Acrobat Reader.

### IBM and accessibility

See the *IBM Accessibility Center* at http://www.ibm.com/able for more information about the IBM commitment to accessibility.

## Dotted decimal syntax diagrams

The syntax diagrams in our publications are available in dotted decimal format, which is an accessible format that is available only if you are using a screen reader.

In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), the elements can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read punctuation. All syntax elements that have the same dotted decimal number (for example, all syntax elements that have the number 3.1) are mutually exclusive

alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, the word or symbol is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is read as 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol that provides information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, that element is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to a separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

**?** Specifies an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element (for example, 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

**!** Specifies a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In

this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

* Specifies a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data-area, you know that you can include more than one data area or you can include none. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

**Notes:**

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

+ Specifies a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times. For example, if you hear the line 6.1+ data-area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. As for the * symbol, you can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on 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 (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.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country 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 and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy,

modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating 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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## 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 http://www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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

# Index

## Special characters



## A



## B



## C



## D

# E



# F



# G



# I

## K



## L



## M



## N



## P



## Q



## R



## S



## T



## U

# V



# W



# X



# Z

Printed in USA

SC27-3518-02

Spine information:

Informix Product Family Data Server Provider for .NET **Version 9.7**

**IBM Data Server Provider for .NET Programmer's Guide, Informix Edition**