

# 1 Overview

InterBase Data Access Components (IBDAC) is a library of components that provides access to InterBase and Firebird database servers. IBDAC directly uses InterBase client software to connect to server. The IBDAC library is designed to help programmers develop faster and cleaner InterBase database applications. IBDAC is a complete replacement for standard InterBase connectivity solutions. It presents an efficient alternative to the Borland Database Engine for access to InterBase and InterBase Express Components.

The IBDAC library is actively developed and supported by the Devart Team. If you have questions about IBDAC, email the developers at [ibdac@devart.com](mailto:ibdac@devart.com) or visit IBDAC online at <http://www.devart.com/ibdac/>.

## Advantages of IBDAC Technology

IBDAC is a direct connectivity database wrapper built specifically for the InterBase server. IBDAC offers wide coverage of the InterBase feature set, and emphasizes optimized data access strategies.

### Wide Coverage of InterBase Features

By providing access to the most advanced database functionality, IBDAC allows developers to harness the full capabilities of the InterBase server and optimize their database applications. IBDAC provides complete support for InterBase Blobs and Arrays, support for Unicode character data, InterBase events. Get a full list of supported InterBase features in the [Features](#) topic.

### Optimized Code

The goal of IBDAC is to enable developers to write efficient and flexible database applications. The IBDAC library is implemented using optimized code and advanced data access algorithms. Component interfaces undergo comprehensive performance tests and are designed to help you write thin and efficient product data access layers. Find out more about how to use IBDAC to optimize your database applications in [Increasing Performance](#).

### Compatibility with other Connectivity Methods

The IBDAC interface retains compatibility with standard VCL data access components like BDE and IBX. Existing BDE- and IBX-based applications can be easily migrated to IBDAC and enhanced to take advantage of InterBase-specific features. Project migration can be automated with the BDE/IBX Migration Wizard. Find out more about Migration Wizard in [Using Migration Wizard](#).

### Development and Support

IBDAC is an InterBase connectivity solution that is actively developed and supported. IBDAC comes with full documentation, demo projects, and fast (usually within one business day) technical support by the IBDAC development team. Find out more about how to get help or submit feedback and suggestions to the IBDAC Development Team in the [Getting Support](#) topic.

A description of the IBDAC components is provided in the [Component List](#).

## Key Features

- Direct access to server data. Does not require installation of other data provider layers (such as BDE and ODBC)

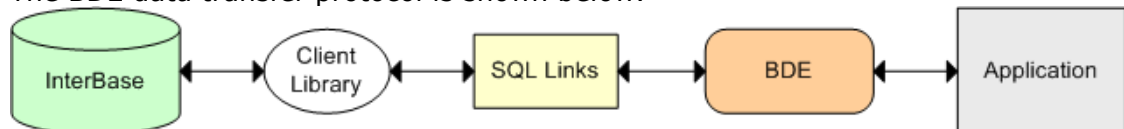
- VCL, LCL and FireMonkey versions of the library available
- Full support of the latest versions of [InterBase and Firebird database servers](#)
- Support for all InterBase data types
- [Disconnected Model](#) with automatic connection control for working with data offline
- [Local Failover](#) for detecting connection loss and implicitly re-executing certain operations
- All types of local [sorting](#) and [filtering](#), including by calculated and lookup fields
- Automatic [data updating](#) with [TIBCQuery](#) and [TIBCTable](#) components
- [Unicode and national charsets support](#)
- [InterBase Events support](#)
- Advanced script execution functionality with [TIBCScript](#) component
- [Support for using Macros in SQL](#)
- Easy migration from BDE and IBX with [Migration Wizard](#)
- Lets you use Professional Edition of [Delphi and C++Builder](#) to develop client/server applications
- Included annual [IBDAC Subscription](#) with [Priority Support](#)
- Licensed royalty-free per developer, per team, or per site

The full list of IBDAC features can be found in the [Features](#) topic.

## How does IBDAC work?

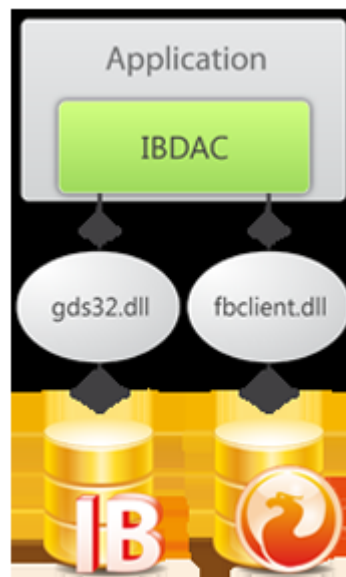
IBDAC directly uses InterBase client software to connect to server. It is designed to be lightweight. It consists of a minimal layer between InterBase server and your code. This extends functionality without sacrificing performance. In comparison, the Borland Database Engine (BDE) uses several layers to access InterBase, and requires additional data access software to be installed on client machines.

The BDE data transfer protocol is shown below.



### BDE Connection Protocol

IBDAC works directly through native InterBase interface. It allows to avoid using BDE or ODBC:



## IBDAC Connection Flow

© 1997-2013 Devart. All Rights Reserved.

## 2 Getting Started

This page contains a quick introduction to setting up and using the InterBase Data Access Components library. It gives a walkthrough of each part of the IBDAC usage process and points out the most relevant related topics in this documentation reference.

- [What is IBDAC?](#)
- [How does IBDAC work?](#)
- [Installing IBDAC.](#)
- [Working with the IBDAC demo projects.](#)
- [Compiling and deploying your IBDAC project.](#)
- [Using the IBDAC documentation.](#)
- [How to get help with IBDAC.](#)

### What is IBDAC?

InterBase Data Access Components (IBDAC) is a component library that provides direct connectivity to InterBase and Firebird for Delphi, C++Builder, and Lazarus (FPC), and helps you develop fast InterBase-based database applications with these environments.

Many IBDAC classes are based on VCL, LCL and FireMonkey classes and interfaces. IBDAC is a replacement for the [Borland Database Engine](#) and InterBase Express, provides native database connectivity, and is specifically designed as an interface to the [InterBase](#) and [Firebird](#) databases.

An introduction to IBDAC is provided in the [Overview](#) section.

A list of the IBDAC features you may find useful is listed in the [Features](#) section.

An overview of the IBDAC component classes is provided in the [Components List](#) section.

## Installing IBDAC

To install IBDAC, complete the following steps.

1. Choose and download the version of the IBDAC installation program that is compatible with your IDE. For instance, if you are installing IBDAC 2.00, you should use the following files:

For BDS 2006 and Turbo - **ibdac200d10\*.exe**

For Delphi 7 - **ibdac200d7\*.exe**

For more information, visit the the [IBDAC download page](#).

2. Close all running IDE's.
3. Launch the IBDAC installation program you downloaded in the first step and follow the instructions to install IBDAC.

By default, the IBDAC installation program should install compiled IBDAC libraries automatically on all IDEs.

To check that IBDAC has been installed properly, launch your IDE and make sure that an InterBase Access page has been added to the Component palette and that an InterBase menu was added to the Menu bar.

If you have bought IBDAC Professional Edition with Source Code, you will be able to download both the compiled version of IBDAC and the IBDAC source code. The installation process for the compiled version is standard, as described above. The IBDAC source code must be compiled and installed manually. Consult the supplied *ReadmeSrc.txt* file for more details.

To find out what gets installed with IBDAC or to troubleshoot your IBDAC installation, visit the [Installation](#) topic.

## Working with the IBDAC demo projects

The IBDAC installation package includes a number of demo projects that demonstrate IBDAC capabilities and use patterns. The IBDAC demo projects are automatically installed in the IBDAC installation folder.

To quickly get started working with IBDAC, launch and explore the introductory IBDAC demo project, *IbDacDemo*, from your IDE. This demo project is a collection of demos that show how IBDAC can be used. The project creates a form which contains an explorer panel for browsing the included demos and a view panel for launching and viewing the selected demo.

### ***IbDacD mo* Walkthrough**

1. Launch your IDE.
2. Choose File Open Project from the menu bar
3. Find the IBDAC directory and open the *IbDacDemo* project. This project should be located in the Demos\IbDacDemo folder.

For example, if you are using Borland Developer Studio 2006, the demo project may be found at

Program Files\Devart\IBDAC for Delphi 2006\Demos\Win32\IbDacDemo  
IbDacDemo.bdsproj

4. Select Run Run or press F9 to compile and launch the demo project. *IbDacDemo* should start, and a full-screen IBDAC Demo window with a toolbar, an explorer panel, and a view panel will open. The explorer panel will contain a list of all the demo sub-projects included in *IbDacDemo*, and the view panel will contain an overview of each included demo.

At this point, you will be able to browse through the available demos, read their descriptions, view their source code, and see the functionality provided

by each demo for interacting with InterBase. However, you will not be able to actually retrieve data from InterBase or execute commands until you connect to the database.

5. Click on the "Connect" button in the *IbDacDemo* toolbar. A Connect dialog box will open. Enter the connection parameters you use to connect to your InterBase server and click "Connect" in the dialog box.

**Note:** For this step to work properly, you must have the InterBase Client installed.

Now you have a fully functional interface to your InterBase server. You will be able to go through the different demos, to browse tables, create and drop objects, and execute DSQL commands.

**Warning!** All changes you make to the database you are connected to, including creating and dropping objects used by the demo, will be permanent. Make sure you specify a test database in the connection step.

6. Click on the "Create" button to create all the objects that will be used by *IbDacDemo*. If some of these objects already exist in the database you have connected to, the following error message will appear.

*"An error has occurred:*

*unsuccessful metadata update Table DEPT already exists*

*You can manually create objects required for demo by using the following file: %IBDAC%\Demos\InstallDemoObjects.sql*

*%IBDAC% is the IBDAC installation path on your computer.*

*Ignore this exception?"*

This is a standard warning from the object execution script. Click "Yes to All" to ignore this message. *IbDacDemo* will create the *IbDacDemo* objects on the server you have connected to.

7. Choose a demo that demonstrates an aspect of working with InterBase that you are interested in, and play with the demo frame in the view window on the right. For example, to find out more about how to work with InterBase tables, select the Table demo from the "Working with Components" folder. A simple InterBase table browser will open in the view panel which will let you open a table in your database by specifying its name and clicking on the Open button.
8. Click on the "Demo source" button in the *IbDacDemo* toolbar to find out how the demo you have selected was implemented. The source code behind the demo project will appear in the view panel. Try to find the places where IBDAC components are used to connect to the database.
9. Click on the "Form as text" button in the *IbDacDemo* toolbar to view the code behind the interface to the demo. Try to find the places where IBDAC components are created on the demo form.
10. Repeat these steps for other demos listed in the explorer window. The available demos are organized in three folders.

### **Working with components**

A collection of projects that show how to work with the basic IBDAC components.

### **General demos**

A collection of projects that show off the IBDAC technology and demonstrate some ways to work with data.

### **InterBase-specific demos**

A collection of projects that demonstrate how to incorporate InterBase/Firebird features in database applications.

11. When you are finished working with the project, click on the "Drop" button in

the *IbDacDemo* toolbar to remove all the schema objects added in Step 6.

### Other IBDAC demo projects

IBDAC is accompanied by a number of other demo projects. A description of all the IBDAC demos is located in the [Demo Projects](#) topic.

## Compiling and deploying your IBDAC project

### Compiling IBDAC-based projects

By default, to compile a project that uses IBDAC classes, your IDE compiler needs to have access to the IBDAC dcu (obj) files. If you are compiling with runtime packages, the compiler will also need to have access to the IBDAC bpl files. **All the appropriate settings for both of these scenarios should take place automatically during the installation of IBDAC.** You should only need to modify your environment manually if you are using one of the IBDAC editions that comes with source code - IBDAC Professional Edition with Source Code.

You can check that your environment is properly configured by trying to compile one of the IBDAC demo projects. If you have no problems compiling and launching the IBDAC demos, your environment is properly configured.

For more information about which library files and environment changes are needed for compiling IBDAC-based projects, consult the [Installation](#) topic.

### Deploying IBDAC-based projects

To deploy an application that uses IBDAC, you will need to make sure the target workstation has access to the following files.

- The InterBase Client software, if connecting in Client mode.
- The IBDAC bpl files, if compiling with runtime packages.
- The IBDAC assembly files, if using VCL for .NET components.

If you are evaluating deploying projects with IBDAC Trial Edition, you will also need to deploy some additional bpl files with your application even if you are compiling without runtime packages. As another trial limitation for C++Builder, applications written with IBDAC Trial Edition for C++Builder will only work if the C++Builder IDE is launched. More information about IBDAC Trial Edition limitations is provided [here](#). A list of the files which may need to be deployed with IBDAC-based applications is included in the [Deployment](#) topic.

## Using the IBDAC documentation

The IBDAC documentation describes how to install and configure IBDAC, how to use IBDAC Demo Projects, and how to use the IBDAC libraries.

The IBDAC documentation includes a detailed reference of all IBDAC components and classes. Many of the IBDAC components and classes inherit or implement members from other VCL, LCL and FireMonkey classes and interfaces. The product documentation also includes a summary of all members within each of these classes. To view a detailed description of a particular component, look it up in the [Components List](#) section. To find out more about a specific standard VCL/LCL class an IBDAC component is inherited from, see the corresponding topic in your IDE documentation.

At install time, the IBDAC documentation is integrated into your IDE. It can be invoked from the InterBase menu added to the Menu Bar, or by pressing F1 in an object inspector or on a selected code segment.

## How to get help with IBDAC

There are a number of resources for finding help on using IBDAC classes in your project.

- If you have any questions about IBDAC installation or licensing, consult the [Licensing](#) and [FAQ](#) sections.
- You can get community assistance and IBDAC technical support on the [IBDAC Support Forum](#).
  - To get help through the IBDAC [Priority Support](#) program, send an email to the IBDAC development team at [ibdac@devart.com](mailto:ibdac@devart.com).
  - If you have any questions about ordering IBDAC or any other Devart product, contact [sales@devart.com](mailto:sales@devart.com).

For more information, consult the [Getting Support](#) topic.

---

© 1997-2013 Devart. All Rights Reserved.

## 3 Features

### General usability:

- Direct access to server data. Does not require installation of other data provider layers (such as BDE and ODBC)
- Interface compatible with standard data access methods, such as BDE and ADO
- VCL, LCL and FireMonkey versions of library available
- [Separated run-time and GUI specific parts](#) allow you to create pure console applications such as CGI
- [Unicode and national charset](#) support

### Network and connectivity:

- [Disconnected Model](#) with automatic connection control for working with data offline
- [Local Failover](#) for detecting connection loss and implicitly reexecuting certain operations

### Compatibility:

- [Full support of the latest versions of InterBase and Firebird database servers](#)
- Support for all InterBase data types
- [Compatible with all IDE versions starting with Delphi 5, C++Builder 5 and La arus \(Free Pascal\)](#)
- Includes provider for UniDAC Standard Edition
- [Wide reporting component support](#), including support for InfoPower,



ReportBuilder, FastReport

- Support of all standard and third-party visual data-aware controls
- Allows you to use Professional Edition of Delphi and C++Builder to develop client/server applications

### **InterBase technology support:**

- Support for fast record insertion with the [TIBCLoader](#) component [New]
- [InterBase event](#) support
- Comprehensive [array data type](#) support
- [Advanced BLOB](#) support
- [Streaming](#) (non-caching) BLOB access support
- [Advanced generator](#) support
- Advanced support for the character set OCTETS
- Support for the Firebird 2 EXECUTE BLOCK syntax
- Support for the Firebird 2 [RETURNING clause](#)
- [Advanced locking](#) for Firebird 2
- [Automatic updates](#) by [DB KEY](#) unique field for Firebird 2
- [Default value support for stored procedures](#)
- InterBase services components for configuring server parameters and security [New]

### **Performance:**

- High overall [performance](#)
- Fast controlled fetch of large data blocks
- Optimized [string data storing](#)
- Advanced [connection pooling](#)
- High performance applying of cached updates with [batches](#)
- [Caching of calculated and lookup fields](#)
- [Fast Locate](#) in a sorted DataSet
- [Preparing of user-defined update statements](#)

### **Local data storage operations:**

- Database-independent data storage with [TVirtualTable](#) component
- [CachedUpdates](#) operation mode
- Local [sorting](#) and filtering, including by calculated and lookup fields
- [Local Master/Detail](#) relationship
- Master/detail relationship in CachedUpdates mode

### **Data access and data management automation:**

- [Automatic data updating](#) with [TIBQuery](#) and [TIBTable](#) components
- Automatic record [refreshing](#) and [locking](#)
- [Automatic query preparing](#)
- Support for ftWideMemo field type in Delphi 2006 and higher

### **Extended data access functionality:**

- [Separate component](#) for executing SQL statements
- Simplified access to table data with [TIBTable](#) component
- [BLOB compression](#) support
- Support for [using macros](#) in SQL
- Ability to customize [update](#) commands by attaching external components to [TIBUpdateSQL](#) objects
- [Deferred detail DataSet refresh](#) in master/detail relationships



- [MIDAS](#) technology support
- [IBCDDataAdapter](#) component for WinForms and ASP.NET applications

**Data exchange:**

- Transferring data between all types of TDataSet descendants with [TCRBatchMove](#) component
- Data [export](#) and [import](#) to/from XML (ADO format)
- Ability to [synchronize positions in different DataSets](#)

**Script execution:**

- Advanced script execution features with [TIBCScript](#) component
- Support for executing [individual statements](#) in scripts
- Support for [executing huge scripts stored in files](#) with dynamic loading
- Ability to use standard ISQL syntax in scripts

**SQL execution monitoring:**

- Extended SQL tracing capabilities provided by [TIBCSQLMonitor](#) component and DBMonitor application
- Borland SQL Monitor support
- Ability to [send messages to DBMonitor](#) from any point in your program

**Visual extensions:**

- Includes source code of enhanced TCRDBGrid data-aware grid control
- Customizable [connection dialog](#)

**Design-time enhancements:**

- DataSet Manager tool to control DataSet instances in the project
- Advanced design-time component and property editors
- Automatic design-time component linking
- Easy migration from BDE and IBX with [Migration Wizard](#)
- More convenient data source setup with the [TIBCDDataSource](#) component
- Syntax highlighting in design-time editors

**Product clarity:**

- Complete documentation sets
- Printable documentation in PDF format
- [A large amount of helpful demo projects](#)

**Licensing and support:**

- Included annual [IBDAC Subscription](#) with [Priority Support](#)
- Licensed royalty-free per developer, per team, or per site

---

© 1997-2013 Devart. All Rights Reserved.

## 4 What's New

**25-Apr-13 New Features in IBDAC 5.0:**

- Rad Studio XE4 is supported
- NEXTGEN compiler is supported
- Application development for iOS is supported

- Connection string support is added
- Possibility to encrypt entire tables and datasets is added
- Possibility to determine if data in a field is encrypted is added
- Support of TimeStamp, Single and Extended fields in VirtualTable is added
- InterBase XE3 ToGo Edition support for iOS device is added
- Additional database shutdown options for TIBCConfigService.ShutdownDatabase are added

### **12-Dec-12 New Features in IBDAC 4.6:**

- Rad Studio XE3 Update 1 is now required
- C++Builder 64-bit for Windows is supported
- TIBCConnection.Port property that allows specifying the port number or the service name for connection is added

### **05-Sep-12 New Features in IBDAC 4.5:**

- Rad Studio XE3 is supported
- Windows 8 is supported

### **23-Nov-11 New Features in IBDAC 4.1:**

- Update 2 for RAD Studio XE2, Delphi XE2, and C++Builder XE2 is now required
- Mac OS X and iOS in RAD Studio XE2 is supported
- FireMonkey support is improved
- La arus 0.9.30.2 and FPC 2.4.4 are supported
- Mac OS X in La arus is supported
- Linux x64 in La arus is supported
- FreeBSD in La arus is supported

### **15-Sep-11 New Features in InterBase Data Access Components 4.00:**

- Embarcadero RAD Studio XE2 is supported
- Application development for 64-bit Windows is supported
- FireMonkey application development platform is supported
- Support of master/detail relationship for TVirtualTable is added
- OnProgress event in TVirtualTable is added
- TDADatasetOptions.SetEmptyStrToNull property that allows inserting NULL value instead of empty string is added
- TIBCDatasetOptions.SetDomainNames property to enable setting TIBCFIELDDESC.DomainName for fields is added
- TIBCLoader.RowsPerBatch property to specify the number of INSERT queries to load in a single batch is added

### **28-Apr-11 New Features in InterBase Data Access Components 3.60:**

- La arus 0.9.30 and FPC 2.4.2 is supported
- TIBCLoader.InsertMode property allowing the use of "UPDATE OR INSERT INTO" syntax for loading data is added
- Possibility to assign Handle to TIBCConnection is added

### **13-Sep-10 New Features in InterBase Data Access Components 3.50:**

- Embarcadero RAD Studio XE supported

## **10-Sep-09 New Features in InterBase Data Access Components 3.10:**

- Embarcadero RAD Studio 2010 supported

## **02-Apr-09 New Features in InterBase Data Access Components 3.00:**

- [TIBCLoader](#) component

serves for fast loading of data to the database. For Firebird 2.0 and higher it combines INSERT statements in one EXECUTE BLOCK statement to speed up loading.

- **InterBase services components**

allow to backup and restore database, configure server parameters and security.

- Free Pascal under Linux supported
- Added NoPreconnect property to TIBCScript for executing CONNECT and CREATE DATABASE commands

## **23-Oct-08 New Features in InterBase Data Access Components 2.70:**

- Delphi 2009 and C++Builder 2009 supported
- Extended Unicode support for Delphi 2007 added (special Unicode build)
- Free Pascal 2.2 supported
- Powerful design-time editors implemented in Lazarus
- Completed with more comprehensive structured Help

## **23-May-08 New Features in InterBase Data Access Components 2.50:**

- Added compatibility with UniDAC
- Improved support of default field values
- The new component for metadata receiving added

## **27-Sep-07 New Features in InterBase Data Access Components 2.20:**

- CodeGear RAD Studio 2007 supported
- Added ability to treat integer fields as TBooleanField when the domain name contains "BOOLEAN"

## **12-Jun-07 New Features in InterBase Data Access Components 2.10:**

- C++Builder 2007 supported

## **22-Mar-07 New Features in InterBase Data Access Components 2.00:**

### **New functionality:**

- Delphi 2007 for Win32 support
- Implemented [Disconnected Model](#) for working offline and automatically connecting and disconnecting

- Implemented [Local Failover](#) for detecting connection loss and implicitly re-executing some operations
- WideMemo field type in Delphi 2006 supported
- Added DataSet Manager to control project datasets
- New [TCRBatchMove](#) component for transferring data between all types of TDataSet descendants added
- Data [export](#) and [import](#) to/from XML supported
- Support for [sending messages](#) to DBMonitor from any point in your program added

#### **Support for more InterBase/Firebird server functionality:**

- [RETURNING clause in the INSERT SQL statement](#) (Firebird 2 server only) supported
- EXECUTE BLOCK syntax (Firebird 2 server only) supported
- Automatic updates by DB Key unique field (Firebird 2 server only) supported
- [Default values in stored procedures](#) supported

#### **Extensions and improvements to existing functionality:**

- General performance improved
- [Master/detail](#) functionality extensions:
  - [Local master/detail](#) relationship support added
  - Support for master/detail relationships in [CachedUpdates](#) mode added
- [Connection pool](#) functionality improvements:
  - Efficiency significantly improved
  - New [API for draining the connection pool](#) added
- [TIBCScript](#) component improvements:
  - Support for executing [individual statements](#) in scripts added
  - Support for [executing huge scripts stored in files](#) with dynamic loading added
  - Ability to use standard ISQL tool syntax added
- Greatly increased [performance of applying updates](#) in [CachedUpdates](#) mode
- Working with [calculated and lookup fields improvements](#):
  - Local [sorting](#) and filtering added
  - Record [location](#) speed increased
  - Improved working with lookup fields
- Ability to customize update commands by attaching external components to [TIBCUpdateSQL](#) objects added
- Ability to [include all fields](#) in automatically generated update SQLs added

#### **Usability improvements:**

- [Syntax highlighting](#) in design-time editors added
- Completely restructured and clearer [demo projects](#)

## **28-Aug-06 New Features in InterBase Data Access Components 1.10:**

- Professional editions of Turbo Delphi, Turbo Delphi for .NET, Turbo C++ supported

## 5 Demo Projects

IBDAC includes a number of demo projects that show off the main IBDAC functionality and development patterns.

The IBDAC demo projects consist of one large project called *IbDacDemo* with demos for all the main IBDAC components, use cases, and data access technologies, and a number of smaller projects on how to use IBDAC in different IDEs and how to integrate IBDAC with third-party components.

Most demo projects are built for Delphi and Borland Developer Studio. There are only two IBDAC demos for C++Builder. However, the C++Builder distribution includes source code for all other demo projects as well.

### Where are the IBDAC demo projects located?

In most cases all the IBDAC demo projects are located in "%IbDac%\Demos\".

In Delphi 2007 under Windows Vista all the IBDAC demo projects are located in "My Documents\Devart\IbDac for Delphi 2007\Demos", for example "C:\Documents and Settings\All Users\Documents\Devart\IbDac for Delphi 2007\Demos\".

The structure of the demo project directory depends on the IDE version you are using.

For most new IDEs with .NET support, the structure will be as follows.

```
Demos
|-.dotNet
|   |-.IbDacDemo [.NET version of the main IBDAC demo project]
|   |-.Miscellaneous
|       |-. [Some other .NET demo projects]
|
|-.Win32
|   |-.IbDacDemo [Win32 version of the main IBDAC demo project]
|   |-.ThirdParty
|       |-. [A collection of demo projects on integration with third-
|           party components]
|       |-.Miscellaneous
|           |-. [Some other Win32 demo projects on design technologies]
```

In Delphi 5, 6, 7, C++Builder 5, 6, and FreePascal .NET is not supported, and the root directories are omitted. For these IDEs you will see the following structure.

```
Demos
|-.IbDacDemo [The main IBDAC demo project]
|-.ThirdParty
|   |-. [A collection of demo projects on integration with third-party
|       components]
|   |-.Miscellaneous
|       |-. [Some other demo projects on design technologies]
```

*IbDacDemo* is the main demo project that shows off all the IBDAC functionality. The other directories contain a number of supplementary demo projects that describe special use cases. A list of all the samples in the IBDAC demo project and a description for the supplementary projects is provided in the following section.

**Note:** This documentation describes ALL the IBDAC demo projects. The actual demo projects you will have installed on your computer depends on your IBDAC version, IBDAC edition, and the IDE version you are using. The integration demos may require installation of third-party components to compile and work properly.

### Instructions for using the IBDAC demo projects

To explore an IBDAC demo project,

1. Launch your IDE.
2. In your IDE, choose File Open Project from the menu bar.
3. Find the directory you've installed IBDAC to and open the Demos folder.
4. Browse through the demo project folders located here and open the project file of the demo you would like to use.
5. Compile and launch the demo. If it exists, consult the *ReadMe.txt* file for more details.

The executed version of the demo will contain a sample application written with IBDAC or a navigable list of samples and sample descriptions. To use each sample properly, you will need to connect to a working InterBase/Firebird server. The included sample applications are fully functional. To use the demos, you have to first set up a connection to InterBase. You can do so by clicking on the "Connect" button.

Many demos may also use some database objects. If so, they will have two object manipulation buttons, "Create" and "Drop". If your demo requires additional objects, click "Create" to create the necessary database objects. When you are done with a demo, click "Drop" to remove all the objects used for the demo from your InterBase database.

**Note:** The IBDAC demo directory includes two sample SQL scripts for creating and dropping all the test database objects used in the IBDAC demos. You can modify and execute this script manually, if you would like. This will not change the behavior of the demos.

You can find a complete walkthrough for the main IBDAC demo project in the [Getting Started](#) topic. Other IBDAC demo projects include a *ReadMe.txt* file with individual building and launching instructions.

## Demo project descriptions

### ***IbDacD mo***

*IbDacDemo* is one large project which includes three collections of demos.

#### **Working with components**

A collection of samples that show how to work with the basic IBDAC components.

#### **General demos**

A collection of samples that show off the IBDAC technology and demonstrate some ways to work with data.

#### **InterBase-specific demos**

A collection of samples that demonstrate how to incorporate InterBase/Firebird features in database applications.

*IbDacDemo* can be opened from %IbDac%\Demos\IbDacDemo\ibdacdemo.dpr (.bdsproj). The following table describes all the demos contained in this project.

## Working with Components

Name	Description
<b>Alerter</b>	Uses the <a href="#">TIBCAlerter</a> component to send messages between connections using InterBase events.
<b>ConnectDialog</b>	Demonstrates how to customize the <a href="#">IBDAC connect dialog</a> . Changes the standard IBDAC connect dialog to two custom connect dialogs. The first customized sample dialog is inherited from the TForm class, and the second one is inherited from the default IBDAC connect dialog class.

<b>CRDBGrid</b>	Demonstrates how to work with the TCRDBGrid component. Shows off the main TCRDBGrid features, like filtering, searching, stretching, using compound headers, and more.
<b>Query</b>	Demonstrates working with <a href="#">TIBCQuery</a> , which is one of the most useful IBDAC components. Includes many TIBCQuery usage scenarios. Demonstrates how to execute queries, how to edit data and export it to XML files. Note: This is a very good introductory demo. We recommend starting here when first becoming familiar with IBDAC.
<b>Sql</b>	Uses <a href="#">TIBCSQL</a> to execute SQL statements. Demonstrates how to prepare the statement and how to work with parameters in SQL.
<b>StoredProc</b>	Uses <a href="#">TIBCStoredProc</a> to access an editable recordset retrieved by an InterBase stored procedure in the client application.
<b>Table</b>	Demonstrates how to use <a href="#">TIBCTable</a> to work with data from a single table on the server without writing any SQL queries manually. Performs server-side data sorting and filtering and retrieves results for browsing and editing.
<b>UpdateSQL</b>	Demonstrates using the <a href="#">TIBCUUpdateSQL</a> component to customize update commands. Lets you optionally use <a href="#">TIBCSQL</a> and <a href="#">TIBCQuery</a> objects for carrying out insert, delete, query, and update commands.
<b>VirtualTable</b>	Demonstrates working with the <a href="#">TVirtualTable</a> component. This sample shows how to fill virtual dataset with data from other datasets, filter data by a given criteria, locate specified records, perform file operations, and change data and table structure.

## General Demos

Name	Description
<b>CachedUpdates</b>	Demonstrates how to perform the most important tasks of working with data in <a href="#">CachedUpdates</a> mode, including highlighting uncommitted changes, managing transactions, and committing changes in a batch.
<b>FilterAndIndex</b>	Demonstrates IBDAC's local storage functionality. This sample shows how to perform local filtering, <a href="#">sorting</a> and <a href="#">locating</a> by multiple fields, including by calculated and lookup fields.
<b>MasterDetail</b>	Uses IBDAC functionality to <a href="#">work with master/detail relationships</a> . This sample shows how to use <a href="#">local master/detail</a> functionality. Demonstrates different kinds of master/detail linking, including linking by SQL, by simple fields, and by calculated fields.
<b>Threads</b>	Demonstrates how IBDAC can be used in multithreaded applications. This sample allows you to set up several threads and test IBDAC's performance with multithreading.

## InterBase-specific Demos

Name	Description
------	-------------



<b>Arrays</b>	Demonstrates <a href="#">working with InterBase arrays</a> . This sample lets you view and control how arrays are represented in dataset fields by the SparseArrays and ObjectView properties.
<b>BlobPictures</b>	Demonstrates <a href="#">working with InterBase BLOB data types</a> . The sample shows how to get binary data from the table. Also it shows off some extended BLOB handling functionality like local caching control, deferred blob reading, getting blob subtype, and more.
<b>DB Key</b>	Demonstrates using Firebird 2.0 RDB\$DB KEY field for building SQLInsert, SQLUpdate and SQLDelete properties.
<b>LongStrings</b>	Demonstrates IBDAC functionality for working with <a href="#">long string fields</a> (fields that have more than 256 characters). Shows the different ways they can be displayed as memo fields and string fields.
<b>TextBlobs</b>	Demonstrates <a href="#">working with InterBase BLOB data types</a> . The sample shows how to get text data from the table. Also it shows off some extended BLOB handling functionality like local caching control, deferred blob reading, blob compression, getting blob subtype, and more.

## Supplementary Demo Projects

IBDAC also includes a number of additional demo projects that describe some special use cases, show how to use IBDAC in different IDEs and give examples of how to integrate it with third-party components. These supplementary IBDAC demo projects are sorted into subfolders in the %IbDac%\Demos\ directory.

Location	Name	Description
<b>dotNet/</b>		<i>[folder appears only for IDEs with support for .NET]</i>
	<b>AspNet</b>	Uses <a href="#">IBCDDataAdapter</a> to create a simple ASP .NET application. This demo shows how to create an ASP.NET application that lets you connect to a database and execute queries. Application displays query results in a DataGrid and sends user changes back to the database.
Miscellaneous	<b>WinForms</b>	Shows how to use IBDAC to create a WinForms application. This demo project creates a simple WinForms application and fills a data grid from an <a href="#">IBCDDataAdapter</a> data source.
<b>IbDacDemo</b>	<b>IbDacDemo</b>	<i>[.NET version of the main IBDAC demo project - see above]</i>
<b>Win32/</b>		<i>[folder appears only for IDEs with support for .NET. For all other IDEs contents appear in root]</i>

	<b>FastReport</b>	Demonstrates how IBDAC can be used with FastReport components. This project consists of two parts. The first part is several packages that integrate IBDAC components into the FastReport editor. The second part is a demo application that lets you design and preview reports with IBDAC technology in the FastReport editor.
	<b>InfoPower</b>	Uses InfoPower components to display recordsets retrieved with IBDAC. This demo project displays an InfoPower grid component and fills it with the result of an IBDAC query. Shows how to link IBDAC data sources to InfoPower components.
ThirdParty	<b>IntraWeb</b>	A collection of sample projects that show how to use IBDAC components as data sources for IntraWeb applications. Contains IntraWeb samples for setting up a connection, querying a database and modifying data, and working with <a href="#">CachedUpdates</a> and <a href="#">MasterDetail</a> relationships.
	<b>Report Builder</b>	Uses IBDAC data sources to create a ReportBuilder report that takes data from InterBase database. Shows how to set up a ReportBuilder document in design-time and how to integrate IBDAC components into the Report Builder editor to perform document design in run-time.

		A general demo project about creating IBDAC-based applications with C++Builder. Lets you execute SQL scripts and work with result sets in a grid. This is one of the two IBDAC demos for C++Builder.
	<b>CBuilder</b>	
	<b>FailOver</b>	Demonstrates the recommended approach to <a href="#">working with unstable networks</a> . This sample lets you perform transactions and updates in several different modes, simulate a sudden session termination, and view what happens to your data state when connections to the server are unexpectedly lost. Shows off CachedUpdates, LocalMasterDetail, FetchAll, Pooling, and different Failover modes.
Miscellaneous	<b>Midas</b>	Demonstrates using MIDAS technology with IBDAC. This project consists of two parts: a MIDAS server that processes requests to the database and a thin MIDAS client that displays an interactive grid. This demo shows how to build thin clients that display interactive components and delegate all database interaction to a server application for processing.
	<b>VirtualTableCB</b>	Demonstrates working with the <a href="#">TVirtualTable</a> component. This sample shows how to fill virtual dataset with data from other datasets, filter data by a given criteria, locate specified records, perform file operations, and change data and table structure. This is one of the two demo projects for C++Builder.
<i>IbDacDemo</i>	<b>IbDacDemo</b>	<i>[Win32 version of the main IBDAC demo project - see above]</i>













©

1997-2013 Devart. All Rights Reserved.









## 6 Component List








This topic presents a brief description of the components included in the InterBase Data Access Components library. Click on the name of each component for more information. These components are added to the InterBase Access page of the Component palette except for [TCRBatchMove](#) and [TVirtualTable](#) components. [TCRBatchMove](#) and [TVirtualTable](#) components are added to the Data Access page of the Component palette.

## Basic IBDAC components

	<a href="#">TIBConnectio n</a>	Sets and controls connection to InterBase database.
	<a href="#">TIBCTransactio n</a>	Provides discrete transaction control over database connections.
	<a href="#">TIBCQuery</a>	Uses SQL statements to retrieve data from InterBase table or tables. Single SELECT statement may be adequately used to generate missing INSERT, DELETE, UPDATE statements.
	<a href="#">TIBCSQL</a>	Executes SQL statements and stored procedures, which do not return rowsets.
	<a href="#">TIBCTable</a>	Lets you retrieve and update data in a single table without writing SQL statements.
	<a href="#">TIBCStoredProc</a>	Executes stored procedures and functions, allows to edit cursor data returned as parameter.
	<a href="#">TIBCUpdateSQ L</a>	Lets you tune update operations for a DataSet component.
	<a href="#">TIBCDataSourc e</a>	Provides an interface between an IBDAC dataset components and data-aware controls on a form.
	<a href="#">TIBCScript</a>	Executes sequences of SQL statements.
	<a href="#">TIBCSQLMonito r</a>	Use to monitor dynamic SQL execution in IBDAC based applications.
	<a href="#">TIBCConnectDi alog</a>	Used to build custom prompts for username, password and server name.
	<a href="#">TVirtualTable</a>	Provides dataset functionality for data that has no real database connection. This component is placed on the Data Access page of the Component palette, not on the InterBase Access page.

## IBDAC Professional Edition components

	<a href="#">TIBCEncryptor</a>	Represents data encryption and decryption in client application.
	<a href="#">TIBCLoader</a>	Allows to load external data into the database table.
	<a href="#">TIBCAlerter</a>	Use to transfer messages between connections.
	<a href="#">TIBCMetaData</a>	Retrieves metadata on specified SQL object.
	<a href="#">TIBCServerPro perties</a>	Returns database server information, including configuration parameters, and also version and license information.
	<a href="#">TIBConfigServ ice</a>	Configures database parameters.
	<a href="#">TIBCLicensingS ervice</a>	Adds or removes InterBase software activation certificates.
	<a href="#">TIBCLogService</a>	Returns the contents of the interbase.log file from server.

	<a href="#">TIBCStatisticalService</a>	Shows database statistics.
	<a href="#">TIBCValidationService</a>	Validates a database and reconciles database transactions.
	<a href="#">TIBCSecurityService</a>	Used to manage user access to the InterBase server.
	<a href="#">TIBCTraceService</a>	Used for working with trace service added in Firebird 2.5.
	<a href="#">TIBCBackupService</a>	Used to backup a database.
	<a href="#">TIBCRestoreService</a>	Used to restore a database.
	<a href="#">TCRBatchMove</a>	Transfers data between all types of TDataSet descendants. This component is placed on the Data Access page of the Component palette, not on the InterBase Access page.

© 1997-2013 Devart. All Rights Reserved.

## 7 Hierarchy Chart

Many IBDAC classes are inherited from standard VCL/LCL classes. The inheritance hierarchy chart for IBDAC is shown below. The IBDAC classes are represented by hyperlinks that point to their description in this documentation. A description of the standard classes can be found in the documentation of your IDE.

```

TObject
|
|--TPersistent
|
|   |--TComponent
|   |
|   |   |--TCustomConnection
|   |   |
|   |   |   |--TCustomDAConnection
|   |   |   |--TIBCCConnection
|   |   |
|   |   |--TDataSet
|   |   |
|   |   |   |--TMemDataSet
|   |   |   |
|   |   |   |   |--TCustomDADataSet
|   |   |   |   |
|   |   |   |   |   |--TCustomIBCDataset
|   |   |   |   |   |
|   |   |   |   |   |   |--TCustomIBCQuery
|   |   |   |   |   |   |
|   |   |   |   |   |   |   |--TIBCQuery
|   |   |   |   |   |   |   |--TIBCStoredProc
|   |   |   |   |   |   |   |--TCustomIBCTable
|   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |--TIBCTable
|   |   |   |   |   |
|   |   |   |   |--TDAMetaData
|   |   |   |   |
|   |   |   |   |   |--TIBCMetaData
|   |   |   |   |
|   |   |   |   |--TVirtualTable
|   |   |
|   |   |--TDataSource
|   |   |
|   |   |   |--TCRDataSource
|   |   |   |--TIBCDDataSource
|   |   |
|   |   |--DADDataAdapter
|   |   |
|   |   |   |--IBCDDataAdapter
|   |   |
|   |   |--TCRBatchMove
|   |   |
|   |   |--TCustomConnectDialog
|   |   |
|   |   |   |--TIBCCConnectDialog
|   |   |
|   |   |--TCustomDASQL

```



© 1997-2013 Devart. All Rights Reserved.

## 9 Editions

IBDAC has two editions: Professional, and Trial. IBDAC Trial Edition is the evaluation version of IBDAC. It includes all the functionality of IBDAC Professional Edition with a trial limitation of 60 days. C++Builder and supported .NET IDEs have additional trial limitations\*.

You can get source code of all the component classes in IBDAC by purchasing the special IBDAC Professional Edition with Source Code\*\*.

For more information about how to get the IBDAC edition you want, visit the [How to Order](#) section.

### IBDAC Edition Matrix

Feature	Professional*	Standard	Trial
<b>Base Components</b>			
<a href="#">TIBConnection</a>			
<a href="#">TIBTransaction</a>			
<a href="#">TIBQuery</a>			
<a href="#">TIBSQL</a>			
<a href="#">TIBTable</a>			
<a href="#">TIBStoredProc</a>	+	+	+
<a href="#">TIBUpdateSQL</a>			
<a href="#">TIBScript</a>			
<a href="#">TIBSQLMonitor</a>			
<a href="#">TIBConnectDialog</a>			
<a href="#">TIBDataSource</a>			
<a href="#">TVirtualTable</a>			
<a href="#">IBDataAdapter</a>			
TCRDBGrid			



### Additional Components

[TIBCEncryptor](#)

[TIBCLoader](#)

[TIBCAlerter](#)

[TIBCMetaData](#)

[TIBCServerProperties](#)

[TIBConfigService](#)

[TIBCLicensingService](#)

+ - +

[TIBLogService](#)

[TIBCStatisticalService](#)

[TIBCBackupService](#)

[TIBCRestoreService](#)

[TIBValidationService](#)

[TIBSecurityService](#)

[TIBCTraceService](#)

[TCRBatchMove](#)

**Design-time features, including component editors and property editors**

+ + +

**[DataSet Manager](#)**\*\*\*

+ - +

**[Migration Wizard](#)**\*\*\*

+ + +

**FreePascal**

+\*\*\*  
\* - +

**Trial limitations\***

- - +

\* *Trial Edition is a fully working version of IBDAC Professional Edition for a trial period of 60 days on most supported IDEs. After the trial period expires you must either register or uninstall IBDAC. IBDAC Trial Edition requires the IDE to be launched on the target workstation when testing .NET applications and applications written on C++Builder. For more information about trial limitations see the [Ordering](#) topic.*

\*\* *Professional Edition with source code is available. Migration Wizard, DataSet Manager source code is not distributed.*

\*\*\* *Not available for C++Builder and FreePascal.*

\*\*\*\* *Available only in editions with source code.*

© 1997-2013 Devart. All Rights Reserved.

## 10 Compatibility

### Database Server Compatibility

IBDAC supports the following database servers:

- InterBase versions since 5.x up to XE3
- Firebird 2.x, 1.x

### IDE Compatibility

IBDAC is compatible with the following IDEs:

- Embarcadero RAD Studio XE4

- Embarcadero Delphi XE4 for Win32
- Embarcadero Delphi XE4 for Win64
- Embarcadero Delphi XE4 for OSX32
- Embarcadero Delphi XE4 for iOS
- Embarcadero C++Builder XE4 for Win32
- Embarcadero C++Builder XE4 for Win64
- Embarcadero C++Builder XE4 for OSX32
- Embarcadero RAD Studio XE3 (Requires [Update 2](#))
  - Embarcadero Delphi XE3 for Win32
  - Embarcadero Delphi XE3 for Win64
  - Embarcadero Delphi XE3 for OSX32
  - Embarcadero C++Builder XE3 for Win32
  - Embarcadero C++Builder XE3 for Win64
  - Embarcadero C++Builder XE3 for OSX32
- Embarcadero RAD Studio XE2 (Requires [Update 4 Hotfix 1](#))
  - Embarcadero Delphi XE2 for Win32
  - Embarcadero Delphi XE2 for Win64
  - Embarcadero Delphi XE2 for OSX32
  - Embarcadero C++Builder XE2 for Win32
  - Embarcadero C++Builder XE2 for OSX32
- Embarcadero RAD Studio XE
  - Embarcadero Delphi XE
  - Embarcadero C++Builder XE
- Embarcadero RAD Studio 2010
  - Embarcadero Delphi 2010
  - Embarcadero C++Builder 2010
- CodeGear RAD Studio 2009 (Requires [Update 3](#))
  - CodeGear Delphi 2009
  - CodeGear C++Builder 2009
- CodeGear RAD Studio 2007
  - CodeGear Delphi 2007 for Win32
  - CodeGear Delphi 2007 for .NET
  - CodeGear C++Builder 2007
- Borland Developer Studio 2006
  - Borland Delphi 2006 for Win32
  - Borland Delphi 2006 for .NET
  - Borland C++Builder 2006
- Turbo Delphi Professional
- Turbo Delphi for .NET Professional
- Turbo C++ Professional
- Borland Delphi 2005
- Borland Delphi 7
- Borland Delphi 6 (Requires [Update Pack 2](#) – Delphi 6 Build 6.240)
- Borland Delphi 5
- Borland C++Builder 6 (Requires [Update Pack 4](#) – C++Builder 6 Build 10.166)
- Borland C++Builder 5
- [La arus](#) 1.0.4 and [Free Pascal](#) 2.6.0 for Windows, Linux, Mac OS X, FreeBSD for 32-bit and 64-bit platforms

Only Architect, Enterprise, and Professional IDE editions are supported. For Delphi XE/XE2/XE3, C++Builder XE/XE2/XE3 IBDAC additionally supports Starter Edition. La arus and Free Pascal are supported only in Trial Edition and Professional Edition with source code.

## Supported Target Platforms

- Windows, 32-bit and 64-bit
- Mac OS X
- iOS
- Linux, 32-bit and 64-bit (only in Lazarus and Free Pascal)
- FreeBSD (only in Lazarus and Free Pascal)

Note that support for 64-bit Windows and Mac OS X was introduced in Rad Studio XE2, and is not available in older versions of Rad Studio. Support for iOS is available in Rad Studio XE2 and XE4, but development for iOS in Rad Studio XE2 is available only with Professional and Developer editions with source code.

## Devart Data Access Components Compatibility

All DAC products are compatible with each other.

But, to install several DAC products to the same IDE, it is necessary to make sure that all DAC products have the same common engine (BPL files) version. The latest versions of DAC products or versions with the same release date always have the same version of the common engine and can be installed to the same IDE.

---

© 1997-2013 Devart. All Rights Reserved.

# 11 Installation

This topic contains the environment changes made by the IBDAC installer. If you are having problems with using IBDAC or compiling IBDAC-based products, check this list to make sure your system is properly configured.

Compiled versions of IBDAC are installed automatically by the IBDAC Installer for all supported IDEs except Lazarus. Versions of IBDAC with Source Code must be installed manually. Installation of IBDAC from sources is described in the supplied *ReadMeSrc.txt* file.

## Before installing IBDAC ...

Two versions of IBDAC cannot be installed in parallel for the same IDE, and, since the Devart Data Access Components products have some shared bpl files, newer versions of IBDAC may be incompatible with older versions of MyDAC, ODAC, and SDAC.

So before installing a new version of IBDAC, uninstall any previous version of IBDAC you may have, and check if your new install is compatible with other Devart Data Access Components products you have installed. For more information please see [Using several products in one IDE](#). If you run into problems or have any compatibility questions, please email [ibdac@devart.com](mailto:ibdac@devart.com)

**Note:** You can avoid performing IBDAC uninstallation manually when upgrading to a new version by directing the IBDAC installation program to overwrite previous versions. To do this, execute the installation program from the command line with a `/force` parameter (Start Run and type `ibdacXX.exe /force`, specifying the full path to the appropriate version of the installation program) .

## Installed packages

The IBDAC package libraries are divided into Win32 project files and .NET project files.

**Note:** %IBDAC% denotes the path to your IBDAC installation directory.

### Delphi/C++Builder Win32 project packages

<i>Name</i>	<i>Description</i>	<i>Location</i>
dacXX.bpl	DAC run-time package	Windows\System32
dcldacXX.bpl	DAC design-time package	Delphi\Bin
dacvclXX.bpl*	DAC VCL support package	Delphi\Bin
ibdacXX.bpl	IBDAC run-time package	Windows\System32
dclibdacXX.bpl	IBDAC design-time package	Delphi\Bin
ibdacvclXX.bpl*	VCL support package	Delphi\Bin
crcontrolsXX.bpl	TCRDBGrid component	Delphi\Bin

\* Not included in Borland Delphi 5 and C++Builder 5. In these IDEs this functionality is distributed among the other packages.

### Delphi for .NET project packages

<i>Name</i>	<i>Description</i>	<i>Location</i>
Devart.Dac.dll	DAC run-time package	Global Assembly Cache
Devart.Dac.Design.dll	DAC design-time package	%IBDAC%\Bin
Devart.Dac.AdoNet.dll	Data provider core package	Delphi\Bin
Devart.IbDac.dll	IBDAC Delphi for .NET run-time package	Global Assembly Cache
Devart.IbDac.Design.dll	IBDAC design-time package	%IBDAC%\Bin
Devart.Vcl.dll	TCRDBGrid component	Global Assembly Cache
Devart.IbDac.AdoNet.dll	Data provider for InterBase package	Global Assembly Cache

### Additional packages for using IBDAC managers and wizards

<i>Name</i>	<i>Description</i>	<i>Location</i>
datasetmanagerXX.bpl	DataSet Manager package	Delphi\Bin
oramigwardXX.dll	IBDAC BDE\IBX Migration wizard	%IBDAC%\Bin

### Additional .NET packages for using IBDAC managers and wizards

<i>Name</i>	<i>Description</i>	<i>Location</i>
Devart.Dac.DsManager.dll	DataSet Manager Assembly	Global Assembly Cache

## Environment Changes

To compile IBDAC-based applications, your environment must be configured to have access to the IBDAC libraries. Environment changes are IDE-dependent. For all instructions, replace %IBDAC% with the path to your IBDAC installation directory

### Delphi

- %IBDAC%\Lib should be included in the Library Path accessible from Tools Environment options Library.

The IBDAC Installer performs Delphi environment changes automatically for compiled versions of IBDAC.

### Delphi for .NET

- Devart.Dac and Devart.IbDac should be included in the Namespace prefixes.
- %IBDAC%\Lib should be included in the Library Path accessible from Tools Options Library - NET.
- %IBDAC%\Bin should be included in the Library Path accessible from Tools Options Library - NET.
- %IBDAC%\Bin should be included in the Component Installed .NET components Assembly Search Path.

The IBDAC Installer performs Delphi for .NET environment changes automatically for compiled versions of IBDAC.

### C++Builder

C++Builder 5, 6:

- \$(BCB)\IBDAC\Lib should be included in the Library Path of the Default Project Options accessible from Project Options Directories/Conditionals.
- \$(BCB)\IBDAC\Include should be included in the Include Path of the Default Project Options accessible from Project Options Directories/Conditionals.

C++Builder 2006, 2007:

- \$(BCB)\IBDAC\Lib should be included in the Library search path of the Default Project Options accessible from Project Default Options C++Builder Linker Paths and Defines.
- \$(BCB)\IBDAC\Include should be included in the Include search path of the Default Project Options accessible from Project Default Options C++Builder C++ Compiler Paths and Defines.

The IBDAC Installer performs C++Builder environment changes automatically for compiled versions of IBDAC.

### Lazarus

The IBDAC installation program only copies IBDAC files. You need to install IBDAC packages to Lazarus IDE manually. Open %IBDAC%\Source\Lazarus1\dcilibdac.lpk file in Lazarus and press the Install button. After that Lazarus IDE will be rebuilt with IBDAC packages.

Do not press the Compile button for the package. Compiling will fail because there are no IBDAC sources.

To check that your environment has been properly configured, try to compile one of the demo projects included with IBDAC. The IBDAC demo projects are located in %IBDAC%\Demos.

## Installation of Additional Components and Add-ins

**DBMonitor**

DBMonitor is a an easy-to-use tool to provide visual monitoring of your database applications. It is provided as an alternative to Borland SQL Monitor that is also supported by IBDAC. DBMonitor is intended to hamper application being monitored as little as possible. For more information, visit the [DBMonitor page online](#).

---

© 1997-2013 Devart. All Rights Reserved.

## 12 Deployment

IBDAC applications can be built and deployed with or without run-time libraries. Using run-time libraries is managed with the "Build with runtime packages" check box in the Project Options dialog box.

### Deploying Win32 applications built without run-time packages

You do not need to deploy any files with IBDAC-based applications built without run-time packages, provided you are using a registered version of IBDAC. You can check if your application does not require run-time packages by making sure the "Build with runtime packages" check box is not selected in the Project Options dialog box.

#### Trial Limitation Warning

If you are evaluating deploying Win32 applications with IBDAC Trial Edition, you will need to deploy the following BPL files and their dependencies (required IDE BPL files) with your application, even if it is built without run-time packages:

dacXX.bpl	always
ibdacXX.bpl	always

### Deploying Win32 applications built with run-time packages

You can set your application to be built with run-time packages by selecting the "Build with runtime packages" check box in the Project Options dialog box before compiling your application.

In this case, you will also need to deploy the following BPL files with your Win32 application:

dacXX.bpl	always
ibdacXX.bpl	always
dacvclXX.bpl	if your application uses the IbDacVcl unit
ibdacvclXX.bpl	if your application uses the IbDacVcl unit
crcontrolsXX.bpl	if your application uses the CRDBGrid component

### Deploying .NET applications

By default you should deploy the following assemblies with your IBDAC .NET application:

Devart.Dac.dll	always
----------------	--------

Devart.IbDac. dll	always
Devart.Dac. AdoNet.dll	If your application uses IBCDataAdapter component
Devart.IbDac. AdoNet.dll	If your application uses IBCDataAdapter component

If you remove the names of these assemblies from the References list of your project, these files will not be required on the target computer.

© 1997-2013 Devart. All Rights Reserved.

## 13 Licensing and Subscriptions

InterBase Data Access Components are licensed, not sold. Please read the end-user license agreement (EULA) carefully before using the product. You can find the EULA in the *License.rtf* file in the IBDAC installation folder.

### Licensing

There are three types of full licenses for IBDAC: Single Licenses, Team Licenses, and Site Licenses.

**Single Licenses** must be purchased for each developer working on a project that uses IBDAC.

Purchasing a **Team License** automatically gives four developers a Single License.

Purchasing a **Site License** automatically gives all developers in a company a Single License.

For evaluation purposes only, you may also use IBDAC Trial Edition under a temporary **Evaluation License**, which allows you to test IBDAC Trial Edition for a period of 60 days, after which you must either remove all files associated with IBDAC or purchase a full license.

To purchase a license for IBDAC, please visit [www.devart.com/ibdac/ordering.html](http://www.devart.com/ibdac/ordering.html).

If you have any questions regarding licensing, please contact [sales@devart.com](mailto:sales@devart.com).

### Editions

Full licenses can be purchased for the following editions of IBDAC: IBDAC Standard Edition, IBDAC Professional Edition, and IBDAC Professional Edition with Source Code.

Users can evaluate IBDAC with IBDAC Trial Edition under Evaluation License.

A comparison chart can be found [here](#).

### Subscriptions

The IBDAC Subscription program is an annual maintenance and support service for IBDAC users.

Users with a valid IBDAC Subscription get the following benefits:

- Product support through the IBDAC [Priority Support](#) program
- Access to new versions of IBDAC when they are released
- Access to all IBDAC updates and bug fixes
- Notification of new product versions

If you have any questions regarding licensing or subscriptions not covered with Help, please contact [sales@devart.com](mailto:sales@devart.com).



## Trial Limitations

IBDAC Evaluation License lets you try IBDAC Trial Edition for a period of 60 days. There are no functionality limitations in IBDAC Trial Edition during the trial period for most supported IDEs, except the following:

- .NET applications and applications written in C++Builder require the corresponding IDE to be launched on the client workstation if they use IBDAC Trial Edition
  - If you are deploying a project built with IBDAC Trial Edition, you will need to include the IBDAC library files in your application deployment package. For more information, consult the [Deployment](#) topic.
- 

© 1997-2013 Devart. All Rights Reserved.

## 14 Getting Support

This page lists several ways you can find help with using IBDAC and describes the IBDAC Priority Support program.

### Support Options

There are a number of resources for finding help on installing and using IBDAC.

- You can find out more about IBDAC installation or licensing by consulting the [Licensing](#) and [FAQ](#) sections.
- You can get community assistance and technical support on the [IBDAC Community Forum](#).
- You can get advanced technical assistance by IBDAC developers through the [IBDAC Priority Support](#) program.

If you have a question about ordering IBDAC or any other Devart product, please contact [sales@devart.com](mailto:sales@devart.com).

### IBDAC Priority Support

IBDAC Priority Support is an advanced product support service for getting expedited individual assistance with IBDAC-related questions from the IBDAC developers themselves. Priority Support is carried out over email and has two business days response policy. Priority Support is available for users with an active [IBDAC Subscription](#).

To get help through the IBDAC Priority Support program, please send an email to [ibdac@devart.com](mailto:ibdac@devart.com) describing the problem you are having. Make sure to include the following information in your message:

- The version of Delphi, C++Builder you are using.
- Your IBDAC Registration number.
- Full IBDAC edition name and version number. You can find both of these in the About sheet of TIBCCConnection Editor or from the InterBase About menu.
- Versions of the InterBase server and client you are using.
- A detailed problem description.

- If possible, a small test project that reproduces the problem. It is recommended to use Scott or SYS schema objects only. Please include definitions for all and avoid using third-party components.

© 1997-2013 Devart. All Rights Reserved.

## 15 Frequently Asked Questions

This page contains a list of Frequently Asked Questions for InterBase Data Access Components.

If you have encounter a question with using IBDAC, please browse through this list first. If this page does not answer your question, refer to the Getting Support topic in IBDAC help

### Installation and Deployment

#### 1. I am having a problem installing IBDAC or compiling IBDAC-based projects...

You may be having a compatibility issue that shows up in one or more of the following forms:

- Get a "Setup has detected already installed DAC packages which are incompatible with current version" message during IBDAC installation.
- Get a "Procedure entry point ... not found in ..." message when starting IDE.
- Get a "Unit ... was compiled with a different version of ..." message on compilation.

You can have such problems if you installed incompatible IBDAC, SDAC, ODAC or MyDAC versions. All these products use common base packages. The easiest way to avoid the problem is to uninstall all installed DAC products and then download from our site and install the last builds.

#### 2. What software should be installed on a client computer for IBDAC-based applications to work?

The minimal configuration of client installation includes the following steps:

- Copy the client file *gds32.dll* to the folder available for executable unit of your program. For example, to the folder with your executable file, or to the Windows system folder. For more information, see description of the *LoadLibrary* function and the environment variable *PATH*.
- Add the "gds db 3050/tcp" line to the *services* file in the %WinDir%\system32\drivers\etc directory.  
For Firebird version 1.0.0.338 and higher, both client and server use port 3050 by default. So, you do not need to modify the *services* file. You can also specify port number for the Firebird client in connection string - *server/3050:c:\dir\data.gdb*
- Copy file *InterBase.msg* (or *firebird.msg* for Firebird) to the folder available for executable unit of your program. File must belong to the same version as InterBase or Firebird.

#### 3. Devart renaming issue that concerns Delphi for .Net users

- Please remove all CoreLab assemblies references from your project and add corresponding Devart ones

- o Please change all unit references in uses clauses from CoreLab to Devart (you can use standard renaming tool)

## Licensing and Subscriptions

### 1. Am I entitled to distribute applications written with IBDAC?

If you have purchased a full version of IBDAC, you are entitled to distribute pre-compiled programs created with its use. You are not entitled to propagate any components inherited from IBDAC or using IBDAC source code. For more information see the *License.rtf* file in your IBDAC installation directory.

### 2. Can I create components using IBDAC?

You can create your own components that are inherited from IBDAC or that use the IBDAC source code. You are entitled to sell and distribute compiled application executables that use such components, but not their source code and not the components themselves.

### 3. What licensing changes can I expect with IBDAC 2.00?

The basic IBDAC license agreement will remain the same. With IBDAC 2.00, the [IBDAC Edition Matrix](#) will be reorganized and a new [IBDAC Subscription Program](#) will be introduced.

### 4. What do the IBDAC 2.00 Edition Levels correspond to?

IBDAC 2.00 will come in three editions: Trial, Professional, and Professional with Sources.

When you upgrade to the new version, your edition level will be automatically updated using the following Edition Correspondence Table.

**Edition Correspondence Table for Upgrading to IBDAC 2.00**

Old Edition Level	New Edition Level
IBDAC Standard Edition	IBDAC Professional Edition
IBDAC Professional Edition	IBDAC Professional Edition with Sources
IBDAC Trial Edition	IBDAC Trial Edition

The feature list for each edition can be found in the IBDAC documentation and on the [IBDAC website](#).

### 5. I have a registered version of IBDAC. Will I need to pay to upgrade to future versions?

After IBDAC 2.00, all upgrades to future versions are free to users with an active IBDAC Subscription.

Users that have a registration for versions of IBDAC prior to IBDAC 2.00 will have to first upgrade to IBDAC 2.00 to jump in on the Subscription program.

### 6. What are the benefits of the IBDAC Subscription Program?

The **IBDAC Subscription Program** is an annual maintenance and support service for IBDAC users.

Users with a valid IBDAC Subscription get the following benefits:

- o Access to new versions of IBDAC when they are released

- o Access to all IBDAC updates and bug fixes
- o Product support through the IBDAC Priority Support program
- o Notification of new product versions

**Priority Support** is an advanced product support program which offers you expedited individual assistance with IBDAC-related questions from the IBDAC developers themselves. Priority Support is carried out over email and has a two business day response policy.

The IBDAC Subscription Program is available for registered users of IBDAC 2.00 and higher.

**7. Can I use my version of IBDAC after my Subscription expires?**

Yes, you can. IBDAC version licenses are perpetual.

**8. I want a IBDAC Subscription! How can I get one?**

An annual IBDAC Subscription is included when ordering or upgrading to any registered (non-Trial) edition of IBDAC 2.00 or higher.

You can renew your IBDAC Subscription on the [IBDAC Ordering Page](#). For more information, please contact [sales@crlab.com](mailto:sales@crlab.com).

**9. Does this mean that if I upgrade to IBDAC 2 from IBDAC 1, I'll get an annual IBDAC Subscription for free?**

Yes.

**10. How do I upgrade to IBDAC 2.00?**

To upgrade to IBDAC 2.00, you can get a Version Update from the [IBDAC Ordering Page](#). For more information, please contact [sales@crlab.com](mailto:sales@crlab.com).

## Performance

**1. How productive is IBDAC?**

IBDAC uses low-level protocol to access the database server. This allows IBDAC to achieve high performance. From time to time we compare IBDAC with other products, and IBDAC always takes first place. For more information refer to [online test results](#).

**2. Why does the Locate function work so slowly the first time I use it?**

Locate is performed on the client. So if you had set FetchAll to False when opening your dataset, cached only some of the rows on the client, and then invoked Locate, IBDAC will have to fetch all the remaining rows from the server before performing the operation. On subsequent calls, Locate should work much faster.

If the Locate method keeps working slowly on subsequent calls or you are working with FetchAll=True, try the following. Perform local sorting by a field that is used in the Locate method. Just assign corresponding field name to the IndexFieldNames property.

## How To

**1. How can I enable syntax highlighting in IBDAC component editors at design time?**

To enable syntax highlighting for IBDAC, you should download and install the freeware [SynEdit component set](#).

**2. How can I determine which version of IBDAC I am using?**

You can determine your IBDAC version number in several ways:

- o During installation of IBDAC, consult the IBDAC Installer screen.
- o After installation, see the *history.html* file in your IBDAC installation directory.
- o At design-time, select InterBase About IBDAC from the main menu of your

IDE.

- o At run-time, check the value of the IbdacVersion and DACVersion constants.

### **3. How can I stop the cursor from changing to an hour glass during query execution?**

Just set the DBAccess.ChangeCursor variable to False anywhere in your program. The cursor will stop changing after this command is executed.

### **4. How can I execute a query saved in the SQLInsert, SQLUpdate, SQLDelete, or SQLRefresh properties of a IBDAC dataset?**

The values of these properties are templates for query statements, and they cannot be manually executed. Usually there is no need to fill these properties because the text of the query is generated automatically.

In special cases, you can set these properties to perform more complicated processing during a query. These properties are automatically processed by IBDAC during the execution of the Post, Delete, or RefreshRecord methods, and are used to construct the query to the server. Their values can contain parameters with names of fields in the underlying data source, which will be later replaced by appropriate data values.

For example, you can use the SQLInsert template to insert a row into a query instance as follows.

- o Fill the SQLInsert property with the parametrized query template you want to use.
- o Call Insert.
- o Initialize field values of the row to insert.
- o Call Post.

The value of the SQLInsert property will then be used by IBDAC to perform the last step.

Setting these properties is optional and allows you to automatically execute additional SQL statements, add calls to stored procedures and functions, check input parameters, and/or store comments during query execution. If these properties are not set, the IBDAC dataset object will generate the query itself using the appropriate insert, update, delete, or refresh record syntax.

### **5. Some questions about the visual part of IBDAC**

The following questions usually arise from the same problem:

- o I set the Debug property to True but nothing happens!
- o While executing a query, the screen cursor does not change to an hour-glass.
- o Even if I have LoginPromp set to True, the connect dialog does not appear.

To fix this problem, you should add the IbDacVcl (for Windows) or IbDacClx (for Linux) unit to the uses clause of your project.

## **General Questions**

### **1. I would like to develop an application that works with InterBase Server. Which should I use - IBDAC or dbExpress?**

dbExpress technology serves for providing a more or less uniform way to access different servers (SQL Server, MySQL, Oracle and so on). It is based on drivers that include server-specific features. Like any universal tool, in many specialized cases dbExpress providers lose some functionality. For example, the dbExpress design-time is quite poor and cannot be expanded.

IBDAC is a specialized set of components to access InterBase server with

advanced design-time and component interface similar to BDE.

We tried to implement maximal InterBase support in IBDAC. dbExpress technology puts severe restrictions. For example, Unicode fields cannot be passed from the driver to dbExpress.

In some cases dbExpress is slower because data undergoes additional conversion to correspond to dbExpress standards.

To summarise, if it is important for you to be able to quickly adapt your application to a database server other than InterBase, it is probably better to use dbExpress. In other cases, especially when migrating from BDE or ADO, you should use IBDAC.

## **2. Why use IBDAC instead of standard InterBase Express components?**

There are many reasons why IBDAC is better than IBExpress. Some of them are enumerated here. For more information refer to IBDAC features list.

- o Reliable user support - we help to solve common issues quickly using e-mail or dedicated forum.
- o IBDAC is being constantly improved and enhanced, so you can be sure that the product is always up-to-date with the latest InterBase data access technology advances.
- o Better support for BLOBs, Arrays and other advanced features of the databases.
- o Automatic generation of SQL UPDATE, INSERT, DELETE, LOCK statements, so that you do not need to care about routine tasks.
- o Ability to lock records automatically, which helps you build stable multiuser applications more easily.
- o Unicode and national charsets support in all IBDAC components
- o IBDAC shares the same troubleproof engine with the other famous DAC products - ODAC, MyDAC, and SDAC. So if you have worked with one of them, it will be easier for you to switch to another one if you ever need to integrate support for another database server in your application.

## **3. Are the IBDAC connection components thread-safe?**

Yes, IBDAC is thread-safe but there is a restriction. The same TIBCCConnection object cannot be used in several threads. So if you have a multithreaded application, you should have a TIBCCConnection object for each thread that uses IBDAC.

## **4. Behaviour of my application has changed when I upgraded IBDAC.**

### **How can I restore the old behaviour with the new version?**

We always try to keep IBDAC compatible with previous versions, but sometimes we have to change behaviour of IBDAC in order to enhance its functionality, or avoid bugs. If either of changes is undesirable for your application, and you want to save the old behaviour, please refer to the "Compatibility with previous versions" topic in IBDAC help. This topic describes such changes, and how to revert to the old IBDAC behaviour.

## **5. When editing a DataSet, I get an exception with the message 'Update failed. Found %d records.' or 'Refresh failed. Found %d records.'**

This error occurs when the database server is unable to determine which record to modify or delete. In other words, there are either more than one record or no records that suit the UPDATE criteria. Such situation can happen when you omit the unique field in a SELECT statement (TCustomDADataset.SQL) or when another user modifies the table simultaneously. This exception

can be suppressed. Refer to TCustomDADataset.Options topic in IBDAC help for more information.

**6.I cannot use INT64 fields as key fields in master-detail relationship.**

Fields of this type are represented in Delphi by TLargeIntField objects. In some versions of Delphi, you cannot access these fields through the Value property (see the protected method TLargeIntField.SetVarValue in the DB unit for details). To avoid this problem, you can change the field type to INT, which is usually sufficient for key fields. Alternatively, you can avoid using Value.

**7.**

**Can IBDAC and BDE functions be used side-by-side in a single application?**

Yes. There is no problem with using both IBDAC and BDE functions in the same application.

---

© 1997-2013 Devart. All Rights Reserved.

## 16 Using IBDAC

### 16.1 Updating Data with IBDAC Dataset Components

Queries are often complex so posting result set modifications to the database becomes not a trivial task. IBDAC dataset components which descend from TCustomIBQuery provide different means for reflecting local changes on the server.

The following components are used to execute SQL statements: TIBQuery, TIBStoredProc, TIBTable.

If application requires result set from a single database table then use TIBTable to query data. Setting only TableName property you may obtain data, modify it and then post changes back to the database.

TIBQuery component may return recordsets from different tables and views all in a single query. There is often no reliable way to make automated update of the database having only original SQL statement or particularly only a name of the stored procedure. To solve this problem additional properties are provided: SQLInsert, SQLUpdate and SQLDelete. Set them with SQL statements which will perform corresponding data modifications on behalf of the original statement whenever insert, update or delete operation is called. You may also assign UpdateObject property with the TIBUpdateSQL class instance which holds all updating SQL statements in one place.

TIBQuery can generate SQL statements for the SQLInsert, SQLUpdate and SQLDelete properties based on the original SQL statement. To identify rows which have to be processed when modified data is applied to the database KeyFields property must be assigned with the names of key fields so that the records are uniquely identified.

For the more careful customization of data update operations you can use [InsertObject](#), [ModifyObject](#) and [DeleteObject](#) properties of [TIBUpdateSQL](#) component.

Set the [Transaction](#) property of your DataSet component to the transaction component with ReadCommitted/ReadOnly [IsolationLevel](#) property, and [UpdateTransaction](#) property to the transaction component with ReadCommitted



[IsolationLevel](#) property for the optimal transaction using performance. Borland recommends to start the read-only transaction and commit it with [CommitRetaining](#) on InterBase 7.1. Using transactions in such a way minimizes server load.

In Firebird 2.0 and higher you can use RETURNING clause of INSERT statement to get the inserted values. It can be useful for getting back inserted values if they are changed by BEFORE INSERT trigger. To add returning clause to SQLInsert automatically set [DMLRefresh](#) property to True.

When you use returning clause of statement in the IBCSQL or IBCQuery component, additional out parameters appear after preparing or executing the statement. They contain returned values and have names like "RET " + column name. Column name is a name of the column of returned value.

For example:

Show code

```
INSERT INTO T1 (F1, F2)
VALUES (:F1, :F2)
RETURNING F1, F2
```

After executing or preparing such statement the following out parameters appear: RET F1 and RET F2. They will contain values of F1 and F2 fields.

IBDAC supports working with RDB\$DB KEY field in Firebird 2.0. RDB\$DB KEY is raw record position in database. DB KEY provides DB KEY field that is used when it is included in the SQL explicitly and KeyFields property is not set. This field is represented with [TIBCDbKeyField](#) class. It will be used for building SQLInsert, SQLUpdate and SQLDelete properties. It can speed up your work because DB KEY is even faster than PK.

**Note:** By default Db Key field initialized with Visible = False. You should explicitly create Db Key field to display it.

## See Also

- [TIBCQuery](#)
- [TIBCStoredProc](#)
- [TIBCTable](#)
- [TIBCDbKeyField](#)
- [TCustomIBCDDataSet.DMLRefresh](#)
- [TCustomIBCDDataSet.UpdateTransaction](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 16.2 Master/Detail Relationships

Master/detail (MD) relationship between two tables is a very widespread one. So it is very important to provide an easy way for database application developer to work with it. Let's examine how IBDAC implements this feature.

Suppose we have classic MD relationship between "Department" and "Employee" tables.

"Department" table has field Dept No. Dept No is a primary key.

"Employee" table has a primary key EmpNo and foreign key Dept No that binds "Employee" to "Department".

It is necessary to display and edit these tables.

IBDAC provides two ways to bind tables. First code example shows how to bind two

TCustomIBCDataset components (TIBCQuery or TIBCTable) into MD relationship via parameters.

```
procedure TForm1.Form1Create(Sender: TObject);
var
  Master, Detail: TIBCQuery;
  MasterSource: TDataSource;
begin
  // create master dataset
  Master := TIBCQuery.Create(Self);
  Master.SQL.Text := 'SELECT * FROM Department';
  // create detail dataset
  Detail := TIBCQuery.Create(Self);
  Detail.SQL.Text := 'SELECT * FROM Employee WHERE Dept_No = :Dept_No';
  // connect detail dataset with master via TDataSource component
  MasterSource := TDataSource.Create(Self);
  MasterSource.DataSet := Master;
  Detail.MasterSource := MasterSource;
  // open master dataset and only then detail dataset
  Master.Open;
  Detail.Open;
end;
```

Pay attention to one thing: parameter name in detail dataset SQL must be equal to the field name in the master dataset that is used as foreign key for detail table. After opening detail dataset always holds records with Dept No field value equal to the one in the current master dataset record.

There is an additional feature: when inserting new records to detail dataset it automatically fills foreign key fields with values taken from master dataset.

**Note:** To make this example first you should place TIBCCConnection and TIBCTransaction components on the form, assign them to each other by setting TIBCCConnection.Transaction and TIBCTransaction.Connection properties and provide connection parameters for connection to the sample database 'Employee'.

Now suppose that detail table "Department" foreign key field is named DepLink but not Dept No. In such case detail dataset described in above code example will not autofill DepLink field with current "Department".Dept No value on insert. This issue is solved in second code example.

```
procedure TForm1.Form1Create(Sender: TObject);
var
  Master, Detail: TIBCQuery;
  MasterSource: TDataSource;
begin
  // create master dataset
  Master := TIBCQuery.Create(Self);
  Master.SQL.Text := 'SELECT * FROM Department';
  // create detail dataset
  Detail := TIBCQuery.Create(Self);
  Detail.SQL.Text := 'SELECT * FROM Employee';
  // setup MD
  Detail.MasterFields := 'Dept_No'; // primary key in Department
  Detail.DetailFields := 'DepLink'; // foreign key in Employee
  // connect detail dataset with master via TDataSource component
  MasterSource := TDataSource.Create(Self);
  MasterSource.DataSet := Master;
  Detail.MasterSource := MasterSource;
  // open master dataset and only then detail dataset
  Master.Open;
  Detail.Open;
end;
```

In this code example MD relationship is set up using [MasterFields](#) and [DetailFields](#) properties. Also note that there are no WHERE clause in detail dataset SQL. To defer refreshing of detail dataset while master dataset navigation you can use [DetailDelay](#) option.

Such MD relationship can be local and remote, depending on the [TCustomDADataset.Options.LocalMasterDetail](#) option. If this option is set to True, dataset uses local filtering for establishing master-detail relationship and does not refer to the server. Otherwise detail dataset performs query each time when record is selected in master dataset. Using local MD relationship can reduce server calls number and save server resources. It can be useful for slow connection. [CachedUpdates](#) mode can be used for detail dataset only for local MD relationship. Using local MD relationship is not recommended when detail table contains too many rows, because in remote MD relationship only records that correspond to the current record in master dataset are fetched. So, this can decrease network traffic in some cases.

### See Also

- [TCustomDADataset.Options](#)
- [TMemDataSet.CachedUpdates](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 16.3 Automatic Key Field Value Generation

When editing dataset it is often convenient not to fill key field(s) values manually but generate them automatically. In the most common way an application developer generates a primary key value basing on a previously created generator. There are three ways to do it.

First, application independent way - developer creates an AFTER INSERT trigger that fills the field value. But there he faces the problem with getting a value inserted by the trigger back to dataset.

Second way is custom key field value generation. A developer can fill a key field value in TCustomIBCQuery.BeforePost event handler. But in that case he should manually execute a query and retrieve the generator value. So this way may be useful only if some special value processing is needed.

Third way, using Generator is the most simple one. A developer only needs to specify two properties - and key field values are generated automatically. There is no need to create a trigger or perform custom BeforePost processing.

```
...
IBCQuery.SQL.Text := 'SELECT DepNo, DepName, Location FROM Department';
IBCQuery.KeyFields := 'DEPT_NO';           // key field
IBCQuery.Generator := 'DeptGenerator';     // generator that will generate values
...
```

### See also

- [KeyGenerator](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 16.4 Data Type Mapping

### Overview

**Data Type Mapping** is a flexible and easily customizable gear, which allows mapping between DB types and Delphi field types.

In this article there are several examples, which can be used when working with all supported DBs. In order to clearly display the universality of the Data Type Mapping gear, a separate DB will be used for each example.

### Data Type Mapping Rules

In versions where Data Type Mapping was not supported, IBDAC automatically set correspondence between the DB data types and Delphi field types. In versions with Data Type Mapping support the correspondence between the DB data types and Delphi field types can be set manually.

Here is the example with the numeric type in the following table of an InterBase or Firebird database:

```
CREATE TABLE NUMERIC_TYPES
(
  ID INTEGER NOT NULL PRIMARY KEY,
  VALUE4 NUMERIC(5, 2),
  VALUE5 NUMERIC(10, 4),
  VALUE6 NUMERIC(15, 6)
)
```

And Data Type Mapping should be used so that:

- the numeric fields with Scale=0 in Delphi would be mapped to one of the field types: TSmallintField, TIntegerField or TLargeintField, depending on Precision
- to save precision, the numeric fields with Precision>=10 and Scale <= 4 would be mapped to TBCDField
- and the numeric fields with Scale >= 5 would be mapped to TFMTBCDField.

The above in the form of a table:

InterBase or Firebird data type	Default Delphi field type	Destination Delphi field type
NUMERIC(5,2)	ftFloat	ftFloat
NUMERIC(10,4)	ftFloat	ftBCD
NUMERIC(15,6)	ftFloat	ftFMTBCD

To specify that numeric fields with Precision <= 4 and Scale = 0 must be mapped to ftSmallint, such a rule should be set:

```
var
  DBType: Word;
  MinPrecision: Integer;
  MaxPrecision: Integer;
  MinScale: Integer;
  MaxScale: Integer;
  FieldType: TFieldType;
begin
  DBType      := ibcNumeric;
  MinPrecision := 10;
  MaxPrecision := rlAny;
  MinScale    := 1;
  MaxScale    := 4;
  FieldType   := ftBCD;
```

```
IBConnection.DataTypeMap.AddDBTypeRule(DBType, MinPrecision, MaxPrecision, Min
end;
```

This is an example of the detailed rule setting, and it is made for maximum visualisation. Usually, rules are set much shorter, e.g. as follows:

```
// rule for numeric(5,2)
IBConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 0, 9, 1, rlAny, ftFloat)
// rule for numeric(10,4)
IBConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 10, rlAny, 1, 4, ftBCD);
// rule for numeric(15,6)
IBConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 10, rlAny, 5, rlAny, ftFMTBCD)
```

## Rules order

When setting rules, there can occur a situation when two or more rules that contradict to each other are set for one type in the database. In this case, only one rule will be applied – the one, which was set first.

For example, there is a table in an InterBase or Firebird database:

```
CREATE TABLE NUMERIC_TYPES
(
  ID INTEGER NOT NULL PRIMARY KEY,
  VALUE4 NUMERIC(5, 2),
  VALUE5 NUMERIC(10, 4),
  VALUE6 NUMERIC(15, 6)
)
```

TBCDField should be used for NUMBER(10,4), and TFMTBCDField - for NUMBER(15,6) instead of default fields:

InterBase or Firebird data type	Default Delphi field type	Destination field type
NUMBER(5,2)	ftFloat	ftFloat
NUMBER(10,4)	ftFloat	ftBCD
NUMBER(15,6)	ftFloat	ftFMTBCD

If rules are set in the following way:

```
IBConnection.DataTypeMap.Clear;
IBConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 0, 9, rlAny, rlAny, ftFlo
IBConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 0, rlAny, 0, 4, ftBCD
IBConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 0, rlAny, 0, rlAny, ftFMT
```

it will lead to the following result:

Oracle data type	Delphi field type
NUMBER(5,2)	ftFloat
NUMBER(10,4)	ftBCD
NUMBER(15,6)	ftFMTBCD

But if rules are set in the following way:

```
IBConnection.DataTypeMap.Clear;
IBConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 0, rlAny, 0, rlAny, ftFMT
IBConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 0, rlAny, 0, 4, ftBCD
IBConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 0, 9, rlAny, rlAny, ftFlo
```

it will lead to the following result:

Oracle data type	Delphi field type
------------------	-------------------

NUMBER(5,2)	ftFMTBCD
NUMBER(10,4)	ftFMTBCD
NUMBER(15,6)	ftFMTBCD

This happens because the rule

```
IBCConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 0, rlAny, 0, rlAny, ftFMTBCD);
```

will be applied for the NUMBER fields, whose Precision is from 0 to infinity, and Scale is from 0 to infinity too. This condition is met by all NUMBER fields with any Precision and Scale.

When using Data Type Mapping, first matching rule is searched for each type, and it is used for mapping. In the second example, the first set rule appears to be the first matching rule for all three types, and therefore the ftFMTBCD type will be used for all fields in Delphi.

If to go back to the first example, the first matching rule for the NUMBER(5,2) type is the first rule, for NUMBER(10,4) - the second rule, and for NUMBER(15,6) - the third rule. So in the first example, the expected result was obtained.

So it should be remembered that if rules for Data Type Mapping are set so that two or more rules that contradict to each other are set for one type in the database, the rules will be applied in the specified order.

## Defining rules for Connection and Dataset

Data Type Mapping allows setting rules for the whole connection as well as for each DataSet in the application.

For example, such table is created in SQL Server:

```
CREATE TABLE PERSON
(
    ID INTEGER NOT NULL PRIMARY KEY,
    FIRSTNAME VARCHAR(20),
    LASTNAME VARCHAR(30),
    GENDER_CODE VARCHAR(1),
    BIRTH_DTTM TIMESTAMP
)
```

It is exactly known that the birth dtm field contains birth day, and this field should be ftDate in Delphi, and not ftDateTime. If such rule is set:

```
IBCConnection.DataTypeMap.Clear;
IBCConnection.DataTypeMap.AddDBTypeRule(ibcTimestamp, ftDate);
```

all DATETIME fields in Delphi will have the ftDate type, that is incorrect. The ftDate type was expected to be used for the DATETIME type only when working with the person table. In this case, Data Type Mapping should be set not for the whole connection, but for a particular DataSet:

```
IBCQuery.DataTypeMap.Clear;
IBCQuery.DataTypeMap.AddDBTypeRule(ibcTimestamp, ftDate);
```

Or the opposite case. For example, DATETIME is used in the application only for date storage, and only one table stores both date and time. In this case, the following rules setting will be correct:

```
IBCConnection.DataTypeMap.Clear;
```

```
IBCConnection.DataTypeMap.AddDBTypeRule(IBCTimeStamp, ftDate);
IBCQuery.DataTypeMap.Clear;
IBCQuery.DataTypeMap.AddDBTypeRule(IBCTimeStamp, ftDateTime);
```

In this case, in all DataSets for the DATETIME type fields with the ftDate type will be created, and for IBCQuery - with the ftDateTime type.

The point is that the priority of the rules set for the DataSet is higher than the priority of the rules set for the whole connection. This allows both flexible and convenient setting of Data Type Mapping for the whole application. There is no need to set the same rules for each DataSet, all the general rules can be set once for the whole connection. And if a DataSet with an individual Data Type Mapping is necessary, individual rules can be set for it.

## Rules for a particular field

Sometimes there is a need to set a rule not for the whole connection, and not for the whole dataset, but only for a particular field.

e.g. there is such table in a MySQL database:

```
CREATE TABLE ITEM
(
  ID INTEGER NOT NULL PRIMARY KEY,
  NAME CHAR(50),
  GUID CHAR(38)
)
```

The **guid** field contains a unique identifier. For convenient work, this identifier is expected to be mapped to the TGuidField type in Delphi. But there is one problem, if to set the rule like this:

```
IBCQuery.DataTypeMap.Clear;
IBCQuery.DataTypeMap.AddDBTypeRule(IBCChar, ftGuid);
```

then both **name** and **guid** fields will have the ftGuid type in Delphi, that does not correspond to what was planned. In this case, the only way is to use Data Type Mapping for a particular field:

```
IBCQuery.DataTypeMap.AddFieldRule('GUID', ftGuid);
```

In addition, it is important to remember that setting rules for particular fields has the highest priority. If to set some rule for a particular field, all other rules in the Connection or DataSet will be ignored for this field.

## Ignoring conversion errors

Data Type Mapping allows mapping various types, and sometimes there can occur the problem with that the data stored in a DB cannot be converted to the correct data of the Delphi field type specified in rules of Data Type Mapping or vice-versa. In this case, an error will occur, which will inform that the data cannot be mapped to the specified type.

For example:

Database value	Destination field type	Error
'text value'	ftInteger	String cannot be converted to Integer
1000000	ftSmallint	Value is out of range

15,1

ftInteger

Cannot convert float to integer

But when setting rules for Data Type Mapping, there is a possibility to ignore data conversion errors:

```
IBConnection.DataTypeMap.AddDBTypeRule(IBCVarChar, ftInteger, True);
```

In this case, the correct conversion is impossible. But because of ignoring data conversion errors, Data Type Mapping tries to return values that can be set to the Delphi fields or DB fields depending on the direction of conversion.

Database value	Destination field type	Result	Result description
'text value'	ftInteger	0	0 will be returned if the text cannot be converted to number
1000000	ftSmallint	32767	32767 is the max value that can be assigned to the Smallint data type
15,1	ftInteger	15	15,1 was truncated to an integer value

Therefore ignoring of conversion errors should be used only if the conversion results are expected.

© 1997-2013 Devart. All Rights Reserved.

## 16.5 Data Encryption

IBDAC has built-in algorithms for data encryption and decryption. To enable encryption, you should attach the [TCREncryptor](#) component to the dataset, and specify the encrypted fields. When inserting or updating data in the table, information will be encrypted on the client side in accordance with the specified method. Also when reading data from the server, the components decrypt the data in these fields "on the fly".

For encryption, you should specify the data encryption algorithm (the [EncryptionAlgorithm](#) property) and password (the [Password](#) property). On the basis of the specified password, the key is generated, which encrypts the data. There is also a possibility to set the key directly using the [SetKey](#) method.

When storing the encrypted data, in addition to the initial data, you can also store additional information: the GUID and the hash. (The method is specified in the [TCREncryptor.DataHeader](#) property).

If data is stored without additional information, it is impossible to determine whether the data is encrypted or not. In this case, only the encrypted data should be stored in the column, otherwise, there will be confusion because of the inability to distinguish the nature of the data. Also in this way, the similar source data will be equivalent in the encrypted form, that is not good from the point of view of the information protection. The advantage of this method is the size of the initial data equal to the size of the encrypted data.

To avoid these problems, it is recommended to store, along with the data, the appropriate GUID, which is necessary for specifying that the value in the record is encrypted and it must be decrypted when reading data. This allows you to avoid



confusion and keep in the same column both the encrypted and decrypted data, which is particularly important when using an existing table. Also, when doing in this way, a random initialing vector is generated before the data encryption, which is used for encryption. This allows you to receive different results for the same initial data, which significantly increases security.

The most preferable way is to store the hash data along with the GUID and encrypted information to determine the validity of the data and verify its integrity. In this way, if there was an attempt to falsify the data at any stage of the transmission or data storage, when decrypting the data, there will be a corresponding error generated. For calculating the hash the SHA1 or MD5 algorithms can be used (the [HashAlgorithm](#) property).

The disadvantage of the latter two methods - additional memory is required for storage of the auxiliary information.

As the encryption algorithms work with a certain size of the buffer, and when storing the additional information it is necessary to use additional memory, TCREncryptor supports encryption of string or binary fields only (*ftString*, *ftWideString*, *ftBytes*, *ftVarBytes*, *ftBlob*, *ftMemo*, *ftWideMemo*). If encryption of string fields is used, firstly, the data is encrypted, and then the obtained binary data is converted into hexadecimal format. In this case, data storage requires two times more space (one byte = 2 characters in hexadecimal).

Therefore, to have the possibility to encrypt other data types (such as date, number, etc.), it is necessary to create a field of the binary or BLOB type in the table, and then convert it into the desired type on the client side with the help of data mapping.

It should be noted that the search and sorting by encrypted fields become impossible on the server side. Data search for these fields can be performed only on the client after decryption of data using the *Locate* and [LocateEx](#) methods. Sorting is performed by setting the [TMemDataSet.IndexFieldNames](#) property.

#### Example.

Let's say there is an employee list of an enterprise stored in the table with the following data: full name, date of employment, salary, and photo. We want all these data to be stored in the encrypted form. Write a script for creating the table:

```
CREATE TABLE EMP
(
    EMPNO INTEGER NOT NULL PRIMARY KEY,
    ENAME VARCHAR(2000) CHARACTER SET OCTETS COLLATE OCTETS,
    HIREDATE VARCHAR(200) CHARACTER SET OCTETS COLLATE OCTETS,
    SAL VARCHAR(200) CHARACTER SET OCTETS COLLATE OCTETS,
    FOTO BLOB SUB_TYPE 0
)
```

As we can see, the fields for storage of the textual information, date, and floating-point number are created with the VARBINARY type. This is for the ability to store encrypted information, and in the case of the text field - to improve performance. Write the code to process this information on the client.

```
IBCQuery.SQL.Text := 'SELECT * FROM EMP';
IBCQuery.Encryption.Encryptor := IBCEncryptor;
IBCQuery.Encryption.Fields := 'ENAME, HIREDATE, SAL, FOTO';
IBCEncryptor.Password := '11111';
IBCQuery.DataTypeMap.AddFieldRule ('ENAME', ftString);
IBCQuery.DataTypeMap.AddFieldRule ('HIREDATE', ftDateTime);
IBCQuery.DataTypeMap.AddFieldRule ('SAL', ftFloat);
IBCQuery.Open;
```

© 1997-2013 Devart. All Rights Reserved.

## 16.6 TIBQuery Component

Important feature of TIBQuery is that TIBQuery is able to generate DML statements for updating data on the server itself. Remember that TIBQuery is able to generate statements for updating only one table. By default TIBQuery uses the first table from FROM clause. You should assign table name to UpdatingTable property if another table is used for updating. If you need more complex SQL statements than generated by TIBQuery, use SQLInsert, SQLUpdate, SQLDelete properties to assign any SQL statements. It is not obligatory to assign all the properties, not assigned are still generated automatically.

### See Also

- [TIBQuery](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 16.7 BLOB Data Types

BLOB field can be used to store large amounts of data of different types. For BLOB data type only BLOB IDs (pointers to data) are stored in table columns; actual BLOB data is stored separately. When accessing a BLOB column, it is the ID which is returned, not the value itself. InterBase supports two types of blobs, stream and segmented. Segmented BLOBs are usual InterBase BLOBs and are stored in chunks. Stream BLOBs are stored as a continuous array of data bytes.

IBDAC components provide following features for working with BLOBs:

- Working with BLOBs the same way like with another fields.
- Fetching BLOB fields on demand.
- Streamed BLOBs support.
- Server side access to BLOB fields.
- Setting BLOB Parameter Buffer (BPB) for using BLOB subtype conversion.
- Compressing BLOB data.

IBDAC components support InterBase BLOB datatype. You can retrieve values of BLOB fields using TIBQuery component the same way as you do for another fields. It is possible to control the way BLOB objects are handled while the application fetches records from the database. BLOBs can be fetched either with other fields to the application or on demand. This is determined by [DeferredBlobRead](#) option in TCustomIBCDataset component. Setting TCustomIBCDataset.Options.

DeferredBlobRead to false allows to reduce traffic over the network since BLOBs are only transferred on demand and to use less memory on the client side because returned record sets do not hold contents of BLOB fields.

IBDAC components support InterBase streamed BLOBs. To enable streamed BLOBs handling set [TCustomIBCDataset.Options.StreamedBlobs](#) to True. Setting this option to True makes all edited BLOBs to be saved as streamed BLOBs and all streamed BLOBs to be handled as streamed. Otherwise streamed BLOBs are handled as usual segmented BLOBs and all edited BLOBs are saved as segmented BLOBs. Setting this option to True also allows to use benefits of server side access to BLOB fields.

Set [TCustomIBCDataset.Options.CacheBlobs](#) to False to access streamed BLOB values on server side without caching BLOBs on the client side. Only requested portions of data are fetched in that case. Setting CacheBlobs to False may bring up

the following benefits for time-critical applications: reduced traffic over the network since only required data are fetched, less memory is needed on the client side because BLOB data are not cached on the client side. This feature is available only for streamed BLOBs and only if StreamedBlobs option is set to True. This option doesn't make sense if DeferredBlobRead is set to False because all BLOB values are fetched to the dataset in that case.

IBDAC components provides easy usage of InterBase BLOB subtype conversion, using BLOB filters. It allows to set BLOB parameter buffer (BPB), that is needed whenever a filter will be used when writing to or reading from a BLOB. BPB contains source and target subtype and charset (for text BLOBs). These parameters are set in [TIBCBlob](#) component by setting [TIBCBlob.CharsetID](#), [TIBCBlob.ConversionCharsetID](#), [TIBCBlob.ConversionSubType](#), [TIBCBlob.SubType](#) properties.

In retrieval operations, when you set them before reading BLOB, SubType and Charset properties are considered actual subtype and charset of database BLOB data. Application will get data converted to ConversionSubType subtype and ConversionCharset charset.

In upload operations, SubType and Charset properties mean actual subtype and charset of BLOB data in the application. ConversionSubType and ConversionCharset properties must contain desired subtype and charset to save BLOB to database with. To avoid unwanted conversions do not set these properties, or make sure that Charset equals to ConversionCharset and SubType equals to ConversionSubType. Note that if there is no filter for pair of source and target subtype, no conversion is provided and BLOB data remains unconverted.

Executing TIBCQuery or TIBCSQL with BLOB parameter requires live transaction after execute. To execute such statement you should explicitly start the transaction or set [AutoCommit](#) property to False.

Use `P:Devart.Dac.TDADatasetOptions.CompressBlobMode` for managing BLOB compression. BLOBs can be stored compressed on the client side, on the server side (in database) or on the both sides. By default it has value `cbNone`, that means no compression is provided. Use `cbClient` value to store compressed blobs on client side. This saves client memory. BLOB data is stored unchanged in database, other application can read these BLOBs as usual. If `cbServer` value is used, BLOB data is stored compressed in database. It's decompressed on the client side. This saves server disk space and network traffic. Other application can't process compressed BLOB data as usual. To use compressed BLOB data both on the client and server side use `cbClientServer` value. To use `cbClient`, `cbServer`, `cbClientServer` and `cbNone` constants you should add `MemData` unit to uses clause.

**Note:** Internal compression functions are available in Delphi 2007, Borland Developer Studio 2006, Delphi 2005, and Delphi 7. To use BLOBcompression under Delphi 6, Delphi 5 and C++Builder you should use your own compression functions. To use them set `CompressProc` and `UncompressProc` variables declared in `MemUtils` unit.

Show code

```
type
  TCompressProc = function(dest: IntPtr; destLen: IntPtr; const source:
    IntPtr; sourceLen: longint): longint;
  TUncompressProc = function(dest: IntPtr; destLen: IntPtr; source:
    IntPtr; sourceLen: longint): longint;
var
  CompressProc: TCompressProc;
  UncompressProc: TUncompressProc;
```

You can compress and decompress a single BLOB. To do it set `P:Devart.Dac.TCompressedBlob.Compressed` property. Set it to `True` to compress BLOB data and to `False` to decompress BLOB data.

Note that using compression and decompression operations will raise CPU usage and can reduce application performance.

## See Also

- [TIBCBlob](#)
- [TCompressedBlob](#)
- `P:Devart.Dac.TCompressedBlob.Compressed`
- [TCustomDADataSet.Options](#)
- [TCustomIBCDDataSet.Options](#)
- [TDAParam.ParamType](#)

© 1997-2013 Devart. All Rights Reserved.

## 16.8 Unicode Character Data

Symbolic information in InterBase can be retrieved for the user as a different character encoding according to the query. InterBase supports number of encoding formats including Unicode. IBDAC components support UTF-8 Unicode (Unicode FSS) encoding formats for data fields.

IBDAC allows to represent string data using string and WideString types. You can use [TIBConnection.Options.UseUnicode](#) property to enable this behaviour. This property value affects fields of queries and stored procedures. [TIBConnection.Options.UseUnicode](#) property has no influence to the parameters types of which were set manually.

Suppose that `SIMPLE TYPES` table is created as:

Show code

```
CREATE TABLE SIMPLE_TYPES (
    ID INTEGER NOT NULL,
    F_CHAR CHAR(250),
    F_VARCHAR VARCHAR2(300),
)
```

Suppose we open the following `SELECT` statement in a dataset

Show code

```
SELECT a.* FROM SIMPLE_TYPES a
```

If [TIBConnection.Options.UseUnicode](#) is set to `False` you get the next fields list after dataset is opened:

Show code

```
ID:      TIntegerField
F_CHAR:  TStringField
F_VARCHAR: TStringField
```

When you set [TIBConnection.Options.UseUnicode](#) to `True` the string fields type changes:

Show code

```
ID:      TIntegerField
F_CHAR:  TWideStringField
F_VARCHAR: TWideStringField
```

Fields of TWideStringField type hold rows in UTF-16 Unicode format. To get a value of the fields you can use TWideStringField.Value property. You can use FlatBuffers, LongString, FieldsAsString, TrimFixedChar options of [TCustomIBCDataset](#) which are compatible with [TIBCCConnection.Options.UseUnicode](#).

To use Unicode values as parameters, previously you need to set a value of data type field to ftWideString or ftFixedWideChar for the fields of VARCHAR or CHAR types accordingly. Otherwise after the execution of AsWideString or AsString operation data type field will be ftString by default.

Show code

```
var
  WS: WideString;
begin
  ...
  with IBCQuery1 do begin
    Close;
    SQL.Text:=
      'SELECT * from SIMPLE_TYPES '+
      'WHERE '+
      '  F_CHAR = :F_CHAR';
    Params[0].DataType := ftFixedWideChar;
    Params[0].AsWideString := WS;
    Open;
  ...
```

Note that if table field has charset NONE or OCTETS, no charset conversion is provided and such fields are always fetched and stored as usual TStringField. You can set charset when creating or modifying table.

If parameter has Unicode data type value, assigning value by using AsString property converts String to WideString. And vice versa, if parameter doesn't have Unicode data type value, assigning value by AsWideString property converts WideString into String.

BLOB data type supports string data in UTF-8 Unicode encoding. You can set [TIBCCConnection.Options.UseUnicode](#) property to True and get TMemoField of ftBlob blob type. BLOB must have subtype 1 (text) and Unicode FSS charset to use this feature.

## See Also

- [TIBCCConnection.Options](#)
- [TCustomIBCDataset.Options](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 16.9 ARRAY Data Type

Some problems appear when you need to use large arrays in dataset. As IBDAC creates one field for each array item great number of TField objects is created. As a

result of it the performance decreases. So IBDAC has the limitation and creates fields for first 1000 items. However, you can access all items with TIBCArry object. If such table is created

Show code

```
CREATE TABLE IBDAC_ARRAYS (
    ID          INTEGER NOT NULL,
    CHAR_ARRAY  CHAR(10) [1:5],
    INTEGER_ARRAY INTEGER [1:2,1:5],
    FLOAT_ARRAY FLOAT [0:8,0:2]
);
```

If ComplexArrayFields is False you can access array item using TIBCArryField

Show code

```
Value := TIBCArryField(IBCQuery1.FieldByName('CHAR_ARRAY')).AsArray.GetItemAsString;
```

If ComplexArrayFields is True, to access array items you can call FieldByName method. For example

Show code

```
Value := Query.FieldByName('CHAR_ARRAY[1]').AsString;
```

If ObjectField property is True this code is right

Show code

```
Value := TADTField(Query.FieldByName('INTEGER_ARRAY[1]')).Fields[0].AsInteger;
```

Using TIBCDataset.GetArray you can access to array items through TIBCArry object

Show code

```
Value := Query.GetArray('FLOAT_ARRAY').GetItemAsFloat([5, 2]);
```

It is possible to control the way Array objects are handled while the application fetches records from the database. Arrays can be fetched either with other fields to the application or on demand. This is determined by [DeferredArrayRead](#) option in [TCustomIBCDataset](#) component. Setting [DeferredArrayRead](#) to False allows to reduce traffic over the network since arrays are only transferred on demand and to use less memory on the client side because returned record sets do not hold contents of the array fields.

Set [TCustomIBCDataset.Options.CacheArrays](#) to False to access array values on server side without caching arrays on the client side. Only requested portions of data are fetched in that case. Setting [CacheArrays](#) to False may bring up the following benefits for time-critical applications: reduced traffic over the network since only required data are fetched, less memory is needed on the client side because array data are not cached on client side.

You can use array type for parameters of SQL statements. You need to assign dtIBCArry to TIBCParm.DataType and use [TIBCParm.AsArray](#) property to access array items.

For example:  
Show code

```
var
  IBSQL: TIBSQL;
. . .
IBSQL.SQL.Text := 'insert into IBDAC_ARRAYS (ID, CHAR_ARRAY) Values(:ID, :CHAR';
IBSQL.ParamByName('ID').AsInteger := 50;
with IBSQL.ParamByName('CHAR_ARRAY').AsArray do begin
  TableName := 'IBDAC_ARRAYS';
  ColumnName := 'CHAR_ARRAY';
  DbHandle := IBSQL.Connection.Handle;
  TrHandle := IBSQL.Transaction.Handle;
  GetArrayInfo;
  SetItemAsString([1], 'AA');
  SetItemAsString([2], 'BB');
  SetItemAsString([3], 'CC');
end;
IBSQL.Execute;
```

## See Also

- [TIBCArray](#)
- [TCustomIBCDDataSet.Options](#)
- Arrays demo

© 1997-2013 Devart. All Rights Reserved.

## 16.10 Working in an Unstable Network

The following settings are recommended for working in an unstable network:

```
TCustomDAConnection.Options.LocalFailover = True
TCustomDAConnection.Options.DisconnectedMode = True
TDataSet.CachedUpdates = True
TCustomDADataSet.FetchAll = True
TCustomDADataSet.Options.LocalMasterDetail = True
AutoCommit = True
```

It is recommended to use ReadCommitted or ReadOnly [IsolationLevel](#) of [TCustomIBCDDataSet.Transaction](#). Use [TCustomIBCDDataSet.UpdateTransaction](#) for update operations. If connection has at least one opened transaction, which is not ReadCommittedReadOnly, FailOver does not execute. All ReadCommittedReadOnly transaction are restored with FailOver operation. In Disconnected mode you can work with one ReadWrite transaction, but it is not recommended.

The following settings are recommended for working with BLOB and array fields in an unstable network.

```
TCustomIBCDDataSet.Options.DeferredBlobRead = False;
TCustomIBCDDataSet.Options.DeferredArrayRead = False;
TCustomIBCDDataSet.Options.CacheArrays = True;
TCustomIBCDDataSet.Options.CacheBlobs = True;
TCustomIBCDDataSet.Options.StreamedBlob = False;
```

These settings allow to work with Blobs and Arrays without an active connection. These settings minimize the number of requests to the server. Using [TCustomDAConnection.Options.DisconnectedMode](#) allows DataSet to work without

an active connection. It minimizes server resource usage and reduces connection break probability. I. e. in this mode connection automatically closes if it is not required any more. But every explicit operation must be finished explicitly. That means each explicit connect must be followed by explicit disconnect. Read [Working with Disconnected Mode](#) topic for more information.

Setting the `P:Devart.IbDac.TCustomIBCDataset.FetchAll` property to `True` allows to fetch all data after cursor opening and to close connection. If you are using master/detail relationship, we recommend to set the [LocalMasterDetail](#) option to `True`. It is not recommended to prepare queries explicitly. Use the [CachedUpdates](#) mode for `DataSet` data editing. Use the [TCustomDADataSet.Options.UpdateBatchSize](#) property to reduce the number of requests to the server.

If a connection breaks, a fatal error occurs, and the [OnConnectionLost](#) event will be raised if the following conditions are fulfilled:

- There are no opened and not fetched datasets;
- There are no explicitly prepared datasets or SQLs.

If the user does not refuse suggested `RetryMode` parameter value (or does not use the [OnConnectionLost](#) event handler), IBDAC can implicitly perform the following operations:

```
Connect;  
DataSet.ApplyUpdates;  
DataSet.Open;
```

I.e. when the connection breaks, implicit reconnect is performed and the corresponding operation is reexecuted. We recommend to wrap other operations in transactions and fulfill their reexecuting yourself.

The using of [Pooling](#) in Disconnected Mode allows to speed up most of the operations because of connecting duration reducing.

## See Also

- FailOver demo
- [Working with Disconnected Mode](#)
- [TCustomDAConnection.Options](#)
- [TCustomDAConnection.Pooling](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 16.11 Disconnected Mode

In disconnected mode a connection opens only when it is required. After performing all server calls connection closes automatically until next server call is required. Datasets remain opened when connection closes. Disconnected Mode may be useful for saving server resources and operating in an unstable or expensive network. Drawback of using disconnected mode is that each connection establishing requires some time for authorization. If connection is often closed and opened it can slow down application work. We recommend to use pooling to solve this problem. For additional information see [TCustomDAConnection.Pooling](#).

To enable disconnected mode set [TCustomDAConnection.Options.DisconnectedMode](#) to `True`.

In disconnected mode a connection is opened for executing requests to the server (if it was not opened already) and is closed automatically if it is not required any more. If the connection was explicitly opened (the [Connect](#) method was called or



the [Connected](#) property was explicitly set to True), it does not close until the [Disconnect](#) method is called or the [Connected](#) property is set to False explicitly. The following settings are recommended to use for working in disconnected mode:

```
TDataSet.CachedUpdates = True
TCustomDADataSet.FetchAll = True
TCustomDADataSet.Options.LocalMasterDetail = True
AutoCommit = True
```

It is recommended to use ReadCommitted or ReadOnly IsolationLevel of [IsolationLevel](#) of [TCustomIBCDDataSet.Transaction](#). Use [TCustomIBCDDataSet.UpdateTransaction](#) for update operations. If connection has at least one opened transaction, which is not ReadCommittedReadOnly, FailOver does not execute. All ReadCommittedReadOnly transaction are restored with FailOver operation. In Disconnected mode you can work with one ReadWrite transaction, but it is not recommended.

These settings minimize the number of requests to the server.

## Disconnected mode features

If you perform a query with the P:Devart.IbDac.TCustomIBCDDataSet.FetchAll option set to True, connection closes when all data is fetched if it is not used by someone else. If the FetchAll option is set to false, connection does not close until all data blocks are fetched.

If explicit transaction was started, connection does not close until the transaction is committed or rolled back.

If the query was prepared explicitly, connection does not close until the query is unprepared or its SQL text is changed.

## See Also

- [TCustomDAConnection.Options](#)
- P:Devart.IbDac.TCustomIBCDDataSet.FetchAll
- [Devart.IbDac.TIBCQuery.LockMode](#)
- [TCustomDAConnection.Pooling](#)
- [TCustomDAConnection.Connect](#)
- [TCustomDAConnection.Disconnect](#)
- [Working in unstable network](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 16.12 Increasing Performance

This topic considers basic stages of working with DataSet and ways to increase performance on each of these stages.

### Connect

If your application performs Connect/Disconnect operations frequently, additional performance can be gained using pooling mode (TCustomDAConnection.Pooling = True). It reduces connection reopening time greatly (hundreds times). Such situation usually occurs in web applications.

### Execute

If your application executes the same query several times, you can use the [TCustomDADataset.Prepare](#) method or set the [TDADatasetOptions.AutoPrepare](#) property to increase performance. For example, it can be enabled for Detail dataset in Master/Detail relationship or for update objects in TDAUpdateSQL. The performance gain achieved this way can be anywhere from several percent to several times, depending on the situation.

To execute SQL statements a [TIBCSQL](#) component is more preferable than [TIBCQuery](#). It can give several additional percents performance gain.

If the [TCustomDADataset.Options.StrictUpdate](#) option is set to False, the [RowsAffected](#) property is not calculated and becomes equal zero. This can improve performance of query executing, so if you need to execute many data updating statements at once and you don't mind affected rows count, set this option to False.

## Fetch

In some situations you can increase performance a bit by using [TCustomDADataset.Options.CompressBlobMode](#). Sometimes using [TCustomIBCDataset.Options.DeferredBlobRead](#) and [TCustomIBCDataset.Options.DeferredArrayRead](#) options with [TCustomIBCDataset.Options.CacheBlobs](#) and [TCustomIBCDataset.Options.CacheArrays](#) set to False can give some additional performance because BLOB and array field contents will be read when required.

You can also tweak your application performance by using the following properties of [TCustomDADataset](#) descendants:

- [FetchRows](#)
- [Options.FlatBuffers](#)
- [Options.LongStrings](#)
- [UniDirectional](#)

See the descriptions of these properties for more details and recommendations.

## Navigate

The [Locate](#) function works faster when dataset is locally sorted on KeyFields fields. Local dataset sorting can be set with the [IndexFieldNames](#) property. Performance gain can be large if the dataset contains a large number of rows.

Lookup fields work faster when lookup dataset is locally sorted on lookup Keys.

Setting the [TDADatasetOptions.CacheCalcFields](#) property can improve performance when locally sorting and locating on calculated and lookup fields. It can be also useful when calculated field expressions contain complicated calculations.

Setting the [TDADatasetOptions.LocalMasterDetail](#) option can improve performance greatly by avoiding server requests on detail refreshes. Setting the [TDADatasetOptions.DetailDelay](#) option can be useful for avoiding detail refreshes when switching master DataSet records frequently.

## Update

If your application updates datasets in the CachedUpdates mode, then setting the [TCustomDADataset.Options.UpdateBatchSize](#) option to more than 1 can improve performance several hundred times more by reducing the number of requests to the server.

Specifying update SQL statements in a dataset improves performance because of

omitting SQL statements generation and automatic preparation of internal updating datasets that are created for every kind of update SQL statements.

You can also increase the data sending performance a bit (several percents) by using `Dataset.UpdateObject.ModifyObject`, `Dataset.UpdateObject`, etc. Little additional performance improvement can be reached by setting the [AutoPrepare](#) property for these objects.

---

© 1997-2013 Devart. All Rights Reserved.

## 16.13 Connection Pooling

Connection pooling enables an application to use a connection from a pool of connections that do not need to be reestablished for each use. Once a connection has been created and placed in a pool, an application can reuse that connection without performing the complete connection process.

Using a pooled connection can result in significant performance gains, because applications can save the overhead involved in making a connection. This can be particularly significant for middle-tier applications that connect over a network or for applications that connect and disconnect repeatedly, such as Internet applications.

To use connection pooling set the `Pooling` property of the [TCustomDAConnection](#) component to `True`. Also you should set the [PoolingOptions](#) of the [TCustomDAConnection](#). These options include [MinPoolSize](#), [MaxPoolSize](#), [Validate](#), [ConnectionLifeTime](#). Connections belong to the same pool if they have identical values for the following parameters: [MinPoolSize](#), [MaxPoolSize](#), [Validate](#), [ConnectionLifeTime](#), [Server](#), [Username](#), [Password](#), [Database](#), [TIBConnectionOptions.Charset](#), [TIBConnectionOptions.UseUnicode](#), [TIBConnectionOptions.Role](#), [SQLDialect](#), [Params](#). When a connection component disconnects from the database the connection actually remains active and is placed into the pool. When this or another connection component connects to the database it takes a connection from the pool. Only when there are no connections in the pool, new connection is established.

Connections in the pool are validated to make sure that a broken connection will not be returned for the [TCustomDAConnection](#) component when it connects to the database. The pool validates connection when it is placed to the pool (e. g. when the [TCustomDAConnection](#) component disconnects). If connection is broken it is not placed to the pool. Instead the pool frees this connection. Connections that are held in the pool are validated every 30 seconds. All broken connections are freed. If you set the [PoolingOptions.Validate](#) to `True`, a connection also will be validated when the [TCustomDAConnection](#) component connects and takes a connection from the pool. When some network problem occurs all connections to the database can be broken. Therefore the pool validates all connections before any of them will be used by a [TCustomDAConnection](#) component if a fatal error is detected on one connection.

The pool frees connections that are held in the pool during a long time. If no new connections are placed to the pool it becomes empty after approximately 4 minutes. This pool behaviour is intended to save resources when the count of connections in the pool exceeds the count that is needed by application. If you set the [PoolingOptions.MinPoolSize](#) property to a non-zero value, this prevents the pool from freeing all pooled connections. When connection count in the pool decreases to [MinPoolSize](#) value, remaining connection will not be freed except if they are broken.

The [PoolingOptions.MaxPoolSize](#) property limits the count of connections that can

be active at the same time. If maximum count of connections is active and some TCustomDAConnection component tries to connect, it will have to wait until any of TCustomDAConnection components disconnect. Maximum wait time is 30 seconds. If active connections' count does not decrease during 30 seconds, the [TCustomDAConnection](#) component will not connect and an exception will be raised. You can limit the time of connection's existence by setting the [PoolingOptions.ConnectionLifeTime](#) property. When the [TCustomDAConnection](#) component disconnects, its internal connection will be freed instead of placing to the pool if this connection is active during the time longer than the value of the [PoolingOptions.ConnectionLifeTime](#) property. This property is designed to make load balancing work with the connection pool.

To force freeing of a connection when the [TCustomDAConnection](#) component disconnects, the [RemoveFromPool](#) method of TCustomDAConnection can be used. You can also free all connection in the pool by using the class procedures Clear or AsyncClear of TIBConnectionPoolManager. These procedures can be useful when you know that all connections will be broken for some reason.

It is recommended to use connection pooling with the [DisconnectMode](#) option of the [TCustomDAConnection](#) component set to True. In this case internal connections can be shared between [TCustomDAConnection](#) components. When some operation is performed on the TCustomDAConnection component (for example, an execution of SQL statement) this component will connect using pooled connection and after performing operation it will disconnect. When an operation is performed on another [TCustomDAConnection](#) component it can use the same connection from the pool.

## See Also

- [TCustomDAConnection.Pooling](#)
  - [TCustomDAConnection.PoolingOptions](#)
  - [Working with Disconnected Mode](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 16.14 Macros

Macros help you to change SQL statements dynamically. They allow partial replacement of the query statement by user-defined text. Macros are identified by their names which are then referred from SQL statement to replace their occurrences for associated values.

First step is to assign macros with their names and values to a dataset object. Then modify SQL statement to include macro names into desired insertion points. Prefix each name with & ("at") sign to let IBDAC discriminate them at parse time. Resolved SQL statement will hold macro values instead of their names but at the right places of their occurrences. For example, having the following statement with the TableName macro name:

```
SELECT * FROM &TableName
```

You may later assign any actual table name to the macro value property leaving your SQL statement intact.

```
Query1.SQL.Text := 'SELECT * FROM &TableName';  
Query1.MacroByName('TableName').Value := 'Dept';  
Query1.Open;
```

IBDAC replaces all macro names with their values and sends SQL statement to the server when SQL execution is requested.

Note that there is a difference between using [TMacro AsString](#) and [Value](#) properties. If you set macro with the [AsString](#) property, it will be quoted. For example, the following statements will result in the same result Query1.SQL property value.

```
Query1.MacroByName('StringMacro').Value := ''A string'';  
Query1.MacroByName('StringMacro').AsString := 'A string';
```

Macros can be especially useful in scripts that perform similar operations on different objects. You can use macros that will be replaced with an object name. It allows you to have the same script text and to change only macro values.

You may also consider using macros to construct adaptable conditions in WHERE clauses of your statements.

## See Also

- [TMacro](#)
- [TCustomDADataSet.MacroByName](#)
- [TCustomDADataSet.Macros](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 16.15 Using Several DAC Products in One IDE

UniDAC, ODAC, SDAC, MyDAC, IBDAC, PgDAC, and LiteDAC components use common base packages (for Win32) and assemblies (for .NET) listed below:

Packages:

- dacXX.bpl
- dacvclXX.bpl
- dcldacXX.bpl

Assemblies:

- Devart.Dac.dll
- Devart.Vcl.dll
- Devart.Dac.Design.dll
- Devart.Dac.AdoNet.dll

Note that product compatibility is provided for the current build only. In other words, if you upgrade one of the installed products, it may conflict with older builds of other products. In order to continue using the products simultaneously, you should upgrade all of them at the same time.

---

© 1997-2013 Devart. All Rights Reserved.

## 16.16 Migration Wizard

### NOTE:

Migration Wizard is available only for Delphi IDE and is not available for C++ Builder.

BDE/IBX Migration Wizard allows you to convert your BDE or IBX projects to IBDAC. This wizard replaces BDE/IBX components at the specified project (dfm-and pas-files) to IBDAC.

To convert a project, perform the following steps.

- Select **BDE/IBX Migration Wizard** from **InterBase** menu
- Select **Replace BDE components** or **Replace IBX components** to replace corresponding components with IBDAC and press the Next button.
- Select the location of the files to search - current open project or disc folder.
- If you have selected Disc folder on the previous step, specify the required folder and specify whether to process subfolders. Press the Next button.
- Select whether to make backup (it is highly recommended to make a backup), backup location, and log parameters, and press the Next button. Default backup location is RBackup folder in your project folder.
- Check your settings and press the Finish button to start the conversion operation.
- The project should be saved before conversion. You will be asked before saving it. Click Yes to continue project conversion.

After the project conversion it will be reopened.

The Wizard just replaces all standard BDE/IBX components. Probably you will need to make some changes manually to compile your application successfully.

If some problems occur while making changes, you can restore your project from backup file. To do this perform the following steps.

- Select **BDE/IBX Migration Wizard** from **InterBase** menu
- Select Restore original files from backup and press the Next button.
- Select the backup file. By default it is RExpert.reu file in RBackup folder of your converted project. Press the Next button.
- Check your settings and press the Finish button to start the conversion operation.
- Press **Yes** in the dialog that appeared.

Your project will be restored to its previous state.

---

© 1997-2013 Devart. All Rights Reserved.

## 16.17 Writing GUI Applications with IBDAC

IBDAC GUI part is standalone. This means that to make GUI elements such as SQL cursors, connect form, connect dialog etc. available, you should explicitly include `IbDacVcl` (`IbDacClx` under Linux) unit in your application. This feature is needed for writing console applications.

### ***D I h and C++Builder***

By default IBDAC does not require Forms, Controls and other GUI related units. Only `TIBConnectDialog` and `TIBErrorHandler` require the Forms unit.

---

© 1997-2013 Devart. All Rights Reserved.

## 16.18 Compatibility with Previous Versions

We always try to keep IBDAC compatible with previous versions, but sometimes we have to change the behaviour of IBDAC in order to enhance its functionality, or avoid bugs. This topic describes such changes, and how to revert the old IBDAC behaviour. We strongly recommend not to turn on the old behaviour of IBDAC. Use options described below only if changes applied to IBDAC crashed your existent application.

Values of the options described below should be assigned in the **initialization** section of one of the units in your project.

**DBAccess.SQLGeneratorCompatibility:**

If the manually assigned [RefreshSQL](#) property contains only "WHERE" clause, IBDAC uses the value of the [BaseSQL](#) property to complete the refresh SQL statement. In this situation all modifications applied to the SELECT query by functions [AddWhere](#), [DeleteWhere](#) are not taken into account. This behavior was changed in IBDAC 2.00.0.4. To restore the old behavior, set the BaseSQLOldBehavior variable to True.

**MemDS.SendDataSetChangeEventAfterOpen:**

Starting with IBDAC 2.20.0.11, the DataSetChangeEvent is sent after the dataset gets open. It was necessary to fix a problem with disappeared vertical scrollbar in some types of DB-aware grids. This problem appears only under Windows XP when visual styles are enabled.

To disable sending this event, change the value of this variable to False.

**MemDS.DoNotRaiseExcetionOnUaFail:**

Starting with IBDAC 2.20.0.12, if the [OnUpdateRecord](#) event handler sets the UpdateAction parameter to uaFail, an exception is raised. The default value of UpdateAction is uaFail. So, the exception will be raised when the value of this parameter is left unchanged.

To restore the old behaviour, set DoNotRaiseExcetionOnUaFail to True.

---

© 1997-2013 Devart. All Rights Reserved.

## 16.19 64-bit Development with Embarcadero RAD Studio XE2

### RAD Studio XE2 Overview

RAD Studio XE2 is the major breakthrough in the line of all Delphi versions of this product. It allows deploying your applications both on Windows and Mac OS platforms. Additionally, it is now possible to create 64-bit Windows applications to fully benefit from the power of new hardware. Moreover, you can create visually spectacular applications with the help of the FireMonkey GPU application platform. Its main features are the following:

- Windows 64-bit platform support;
- Mac OS support;
- FireMonkey application development platform;
- Live data bindings with visual components;
- VCL styles for Windows applications.

For more information about RAD Studio XE2, please refer to [World Tour](#).

### Changes in 64-bit Application Development

64-bit platform support implies several important changes that each developer must keep in mind prior to the development of a new application or the modernization of an old one.

**General**

RAD Studio XE2 IDE is a 32-bit application. It means that it cannot load 64-bit packages at design-time. So, all design-time packages in RAD Studio XE2 IDE are 32-bit.



Therefore, if you develop your own components, you should remember that for the purpose of developing components with the 64-bit platform support, you have to compile run-time packages both for the 32- and 64-bit platforms, while design-time packages need to be compiled only for the 32-bit platform. This might be a source of difficulties if your package is simultaneously both a run-time and a design-time package, as it is more than likely that this package won't be compiled for the 64-bit platform. In this case, you will have to separate your package into two packages, one of which will be used as run-time only, and the other as design-time only. For the same reason, if your design-time packages require that certain DLLs be loaded, you should remember that design-time packages can be only 32-bit and that is why they can load only 32-bit versions of these DLLs, while at run-time 64-bit versions of the DLLs will be loaded. Correspondingly, if there are only 64-bit versions of the DLL on your computer, you won't be able to use all functions at design-time and, vice versa, if you have only 32-bit versions of the DLLs, your application won't be able to work at run-time.

#### Extended type

For this type in a 64-bit applications compiler generates SSE2 instructions instead of FPU, and that greatly improves performance in applications that use this type a lot (where data accuracy is needed). For this purpose, the size and precision of Extended type is reduced:

T PE	32-bit	64-bit
Extended	10 bytes	8 bytes

The following two additional types are introduced to ensure compatibility in the process of developing 32- and 64-bit applications:

Extended80 – whose size in 32-bit application is 10 bytes; however, this type provides the same precision as its 8-byte equivalent in 64-bit applications.

Extended80Rec – can be used to perform low-level operations on an extended precision floating-point value. For example, the sign, the exponent, and the mantissa can be changed separately. It enables you to perform memory-related operations with 10-bit floating-point variables, but not extended-precision arithmetic operations.

#### Pointer and Integers

The major difference between 32- and 64-bit platforms is the volume of the used memory and, correspondingly, the size of the pointer that is used to address large memory volumes.

T PE	32-bit	64-bit
Pointer	4 bytes	8 bytes

At the same time, the size of the Integer type remains the same for both platforms:

T PE	32-bit	64-bit
Integer	4 bytes	4 bytes

That is why, the following code will work incorrectly on the 64-bit platform:

```
Ptr := Pointer(Integer(Ptr) + Offset);
```

While this code will correctly on the 64-bit platform and incorrectly on the 32-bit platform:

```
Ptr := Pointer(Int64(Ptr) + Offset);
```

For this purpose, the following platform-dependent integer type is introduced:

T PE	32-bit	64-bit
------	--------	--------



NativeInt	4 bytes	8 bytes
NativeUInt	4 bytes	8 bytes

This type helps ensure that pointers work correctly both for the 32- and 64-bit platforms:

```
Ptr := Pointer(NativeInt(Ptr) + Offset);
```

However, you need to be extra-careful when developing applications for several versions of Delphi, in which case you should remember that in the previous versions of Delphi the NativeInt type had different sizes:

TYPE	Delphi Version	Size
NativeInt	D5	N/A
NativeInt	D6	N/A
NativeInt	D7	8 bytes
NativeInt	D2005	8 bytes
NativeInt	D2006	8 bytes
NativeInt	D2007	8 bytes
NativeInt	D2009	4 bytes
NativeInt	D2010	4 bytes
NativeInt	Delphi XE	4 bytes
NativeInt	Delphi XE2	4 or 8 bytes

### Out parameters

Some WinAPIs have OUT parameters of the SIZE\_T type, which is equivalent to NativeInt in Delphi XE2. The problem is that if you are developing only a 32-bit application, you won't be able to pass Integer to OUT, while in a 64-bit application, you will not be able to pass Int64; in both cases you will have to pass NativeInt. For example:

```
procedure MyProc(out Value: NativeInt);
begin
    Value := 12345;
end;
var
    Value1: NativeInt;
{$IFDEF WIN32}
    Value2: Integer;
{$ENDIF}
{$IFDEF WIN64}
    Value2: Int64;
{$ENDIF}
begin
    MyProc(Value1); // will be compiled;
    MyProc(Value2); // will not be compiled !!!
end;
```

### Win API

If you pass pointers to SendMessage/PostMessage/TControl.Perform, the wParam and lParam parameters should be type-casted to the WPARAM/LPARAM type and not to Integer/Longint.

#### Correct:

```
SendMessage(hWnd, WM_SETTEXT, 0, LPARAM(@MyCharArray));
```

Wrong:

```
SendMessage(hWnd, WM_SETTEXT, 0, Integer(@MyCharArray));
```

Replace SetWindowLong/GetWindowLong with SetWindowLongPtr/GetWindowLongPtr for GWLP\_HINSTANCE, GWLP\_ID, GWLP\_USERDATA, GWLP\_HWNDPARENT and GWLP\_WNDPROC as they return pointers and handles. Pointers that are passed to SetWindowLongPtr should be type-casted to LONG\_PTR and not to Integer/Longint.

Correct:

```
SetWindowLongPtr(hWnd, GWLP_WNDPROC, LONG_PTR(@MyWindowProc));
```

Wrong:

```
SetWindowLong(hWnd, GWL_WNDPROC, Longint(@MyWindowProc));
```

Pointers that are assigned to the TMessage.Result field should use a type-cast to LRESULT instead of Integer/Longint.

Correct:

```
Message.Result := LRESULT(Self);
```

Wrong:

```
Message.Result := Integer(Self);
```

All TWM...-records for the windows message handlers must use the correct Windows types for the fields:

```
Msg: UINT; wParam: WPARAM; lParam: LPARAM; Result: LRESULT)
```

**Assembler**

In order to make your application (that uses assembly code) work, you will have to make several changes to it:

- rewrite your code that mixes Pascal code and assembly code. Mixing them is not supported in 64-bit applications;
- rewrite assembly code that doesn't consider architecture and processor specifics. You can use conditional defines to make your application work with different architectures.

You can learn more about Assembly code here: [http://docwiki.embarcadero.com/RADStudio/en/Using\\_Inline\\_Assembly\\_Code](http://docwiki.embarcadero.com/RADStudio/en/Using_Inline_Assembly_Code) You can also look at the following article that will help you to make your application support the 64-bit platform: [http://docwiki.embarcadero.com/RADStudio/en/Converting\\_32-bit\\_Delphi\\_Applications\\_to\\_64-bit\\_Windows](http://docwiki.embarcadero.com/RADStudio/en/Converting_32-bit_Delphi_Applications_to_64-bit_Windows)

**Exception handling**

The biggest difference in exception handling between Delphi 32 and 64-bit is that in Delphi XE2 64-bit you will gain more performance because of different internal exception mechanism. For 32-bit applications, the Delphi compiler (dcc32.exe) generates additional code that is executed any way and that causes performance loss. The 64-bit compiler (dcc64.exe) doesn't generate such code, it generates metadata and stores it in the PDATA section of an executable file instead. But in Delphi XE2 64-bit it's impossible to have more than 16 levels of nested exceptions. Having more than 16 levels of nested exceptions will cause a Run Time error.

**Debugging**

Debugging of 64-bit applications in RAD Studio XE2 is remote. It is caused by the same reason: RAD Studio XE2 IDE is a 32 application, but your application is 64-bit. If you are trying to debug your application and you cannot do it, you should check that the **Include remote debug symbols** project option is enabled.

To enable it, perform the following steps:

1. Open Project Options (in the main menu **Project->Options**).
2. In the Target combobox, select **Debug configuration - 64-bit Windows platform**. If there is no such option in the combobox, right click "Target Platforms" in Project Manager and select **Add platform**. After adding the 64-bit Windows platform, the **Debug configuration - 64-bit Windows platform** option will be available in the Target combobox.
3. Select **Linking** in the left part of the Project Options form.
4. enable the **Include remote debug symbols** option.

After that, you can run and debug your 64-bit application.

To enable remote debugging, perform the following steps:

1. Install Platform Assistant Server (PAServer) on a remote computer. You can find PAServer in the %RAD Studio XE2 Install Directory%\PAServer directory. The setup paserver.exe file is an installation file for Windows, and the setup paserver. ip file is an installation file for MacOS.
2. Run the PAServer.exe file on a remote computer and set the password that will be used to connect to this computer.
3. On a local computer with RAD Studio XE2 installed, right-click the target platform that you want to debug in Project Manager and select **Assign Remote Profile**. Click the **Add** button in the displayed window, input your profile name, click the **Next** button, input the name of a remote computer and the password to it (that you assigned when you started PAServer on a remote computer).

After that, you can test the connection by clicking the **Test Connection** button. If your connection failed, check that your firewalls on both remote and local computers do not block your connection, and try to establish a connection once more. If your connection succeeded, click the Next button and then the Finish button. Select your newly created profile and click **OK**.

After performing these steps you will be able to debug your application on a remote computer. Your application will be executed on a remote computer, but you will be able to debug it on your local computer with RAD Studio XE2.

For more information about working with Platform Assistant Server, please refer to [http://docwiki.embarcadero.com/RADStudio/en/Installing\\_and\\_Running\\_the\\_Platform\\_Assistant\\_on\\_the\\_Target\\_Platform](http://docwiki.embarcadero.com/RADStudio/en/Installing_and_Running_the_Platform_Assistant_on_the_Target_Platform)

---

© 1997-2013 Devart. All Rights Reserved.

## 16.20 Database Specific Aspects of 64-bit Development

### InterBase and FireBird Connectivity Aspects

To work with InterBase and Firebird, IBDAC uses their client libraries (gds32.dll and fbclient.dll correspondingly). If you are developing a 32-bit application, then the development process will not be different for you, except some peculiarities of each particular platform. But if you are developing a 64-bit application, you have to be aware of specifics of working with client libraries at design-time and run-time. To connect to an InterBase or Firebird database at design-time, you must have its 32-bit client library. You have to place it to the C:\Windows\SysWOW64 directory. This requirement flows out from the fact that RAD Studio XE2 is a 32-bit application and it cannot load 64-bit libraries in design-time. To work with an InterBase or Firebird database at run-time (64-bit application), you must have the 64-bit client library placed to the C:\Windows\System32 directory.

© 1997-2013 Devart. All Rights Reserved.

## 17 Reference

This page shortly describes units that exist in IBDAC.

### Units

Unit Name	Description
<a href="#">CRAccess</a>	This unit contains base classes for accessing databases.
<a href="#">CRBatchMove</a>	This unit contains implementation of the TCRBatchMove component.
<a href="#">CRDataTypeMap</a>	This unit contains base classes for Data Type Mapping
<a href="#">CREncryption</a>	This unit contains base classes for data encryption.
<a href="#">DAAlerter</a>	This unit contains the base class for the TIBCAlerter component.
<a href="#">DADump</a>	This unit contains the base class for the TIBCDump component.
<a href="#">DALoader</a>	This unit contains the base class for the TIBCLoader component.
<a href="#">DAScript</a>	This unit contains the base class for the TIBCSript component.
<a href="#">DASQLMonitor</a>	This unit contains the base class for the TIBCSQLMonitor component.
<a href="#">DBAccess</a>	This unit contains base classes for most of the components.
<a href="#">Devart.Dac.DataAdapter</a>	This unit contains implementation of the DADDataAdapter class.
<a href="#">Devart.IbDac.DataAdapter</a>	This unit contains implementation of the IBIDataAdapter class.
<a href="#">IBC</a>	This unit contains main components of IBDAC.

<a href="#">IBCAAdmin</a>	This unit contains implementation of components, used for InterBase/Firebird server administration.
<a href="#">IBCAlerter</a>	This unit contains implementation of the TIBCAlerter component.
<a href="#">IBCArray</a>	This unit contains the TIBCArray class for representing the value of the InterBase array data type.
<a href="#">IBCCall</a>	Defines InterBase Call Interface routines.
<a href="#">IBCClasses</a>	IBCClasses unit defines the following data type constants: dtDbKey dtFixedChar dtFixedWideChar
<a href="#">IBCConnectionPool</a>	This unit contains the TIBCConnectionPoolManager class for managing connection pool.
<a href="#">IBCErrors</a>	IBCErrors unit implements the following classes: <a href="#">EIBCErrors</a> .
<a href="#">IBCLoader</a>	This unit contains implementation of the TIBCLoader component.
<a href="#">IBCScript</a>	This unit contains implementation of the TIBCScript component.
<a href="#">IBCSQLMonitor</a>	This unit contains implementation of the TIBCSQLMonitor component.
<a href="#">IbDacVcl</a>	This unit contains the visual constituent of IBDAC.
<a href="#">MemData</a>	This unit contains classes for storing data in memory.
<a href="#">MemDS</a>	This unit contains implementation of the TMemDataSet class.
<a href="#">MemUtils</a>	This unit contains auxiliary procedures and functions used in the DAC code.
<a href="#">VirtualTable</a>	This unit contains implementation of the TVirtualTable component.

© 1997-2013 Devart. All Rights Reserved.

## 17.1 CRAccess

This unit contains base classes for accessing databases.

### Classes

Name	Description
<a href="#">TCRCursor</a>	A base class for classes that work with database cursors.

### Types

Name	Description
<a href="#">TBeforeFetchProc</a>	This type is used for the <a href="#">TCustomDADataset.BeforeFetch</a> event.

### Enumerations

Name	Description
<a href="#">TCRIsolationLevel</a>	Specifies how to handle transactions containing database modifications.
<a href="#">TCRTransactionAction</a>	Specifies the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.1.1 Classes

Classes in the **CRAccess** unit.

#### Classes

Name	Description
<a href="#">TCRCursor</a>	A base class for classes that work with database cursors.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.1.1.1 TCRCursor Class

A base class for classes that work with database cursors.  
For a list of all members of this type, see [TCRCursor](#) members.

#### Unit

[CRAccess](#)

## Syntax

```
TCRCursor = class (TSharedObject) ;
```

## Remarks

TCRCursor is a base class for classes that work with database cursors.

## Inheritance Hierarchy

[TSharedObject](#)  
**TCRCursor**

© 1997-2013 Devart. All Rights Reserved.

### 17.1.1.1.1 Members

[TCRCursor](#) class overview.

## Properties

Name	Description
<a href="#">RefCount</a> (inherited from <a href="#">TSharedObject</a> )	Used to return the count of reference to a TSharedObject object.

## Methods

Name	Description
<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">Release</a> (inherited from <a href="#">TSharedObject</a> )	Decrements the reference count.

© 1997-2013 Devart. All Rights Reserved.

## 17.1.2 Types

Types in the **CRAccess** unit.

## Types

Name	Description
<a href="#">TBeforeFetchProc</a>	This type is used for the <a href="#">TCustomDADataset.BeforeFetch</a> event.

© 1997-2013 Devart. All Rights Reserved.

### 17.1.2.1 TBeforeFetchProc Procedure Reference

This type is used for the [TCustomDADataset.BeforeFetch](#) event.

## Unit

[CRAccess](#)**Syntax**

```
TBeforeFetchProc = procedure (var Cancel: boolean) of object;
```

**Parameters***Cancel*

True, if the current fetch operation should be aborted.

© 1997-2013 Devart. All Rights Reserved.

**17.1.3 Enumerations**Enumerations in the **CRAccess** unit.**Enumerations**

<b>Name</b>	<b>Description</b>
<a href="#">TCRIsolationLevel</a>	Specifies how to handle transactions containing database modifications.
<a href="#">TCRTransactionAction</a>	Specifies the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.

© 1997-2013 Devart. All Rights Reserved.

**17.1.3.1 TCRIsolationLevel Enumeration**

Specifies how to handle transactions containing database modifications.

**Unit**[CRAccess](#)**Syntax**

```
TCRIsolationLevel = (ilReadCommitted);
```

**Values**

<b>Value</b>	<b>Meaning</b>
<b>ilReadCommitted</b>	The default transaction behavior. If the transaction contains DML that requires row locks held by another transaction, then the DML statement waits until the row locks are released.

© 1997-2013 Devart. All Rights Reserved.



### 17.1.3.2 TCRTransactionAction Enumeration

Specifies the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.

#### Unit

[CRAccess](#)

#### Syntax

```
TCRTransactionAction = (taCommit, taRollback);
```

#### Values

Value	Meaning
<b>taCommit</b>	Transaction is committed.
<b>taRollback</b>	Transaction is rolled back.

© 1997-2013 Devart. All Rights Reserved.

## 17.2 CRBatchMove

This unit contains implementation of the TCRBatchMove component.

#### Classes

Name	Description
<a href="#">TCRBatchMove</a>	Transfers records between datasets.

#### Types

Name	Description
<a href="#">TCRBatchMoveProgressEvent</a>	This type is used for the <a href="#">TCRBatchMove.OnBatchMoveProgress</a> event.

#### Enumerations

Name	Description
<a href="#">TCRBatchMode</a>	Used to set the type of the batch operation that will be executed after calling the <a href="#">TCRBatchMove.Execute</a> method.
<a href="#">TCRFieldMappingMode</a>	Used to specify the way fields of the destination and source datasets will be mapped to each other if the <a href="#">TCRBatchMove.Mappings</a> list is empty.

© 1997-2013 Devart. All Rights Reserved.

## 17.2.1 Classes

Classes in the **CRBatchMove** unit.

### Classes

Name	Description
<a href="#">TCRBatchMove</a>	Transfers records between datasets.

© 1997-2013 Devart. All Rights Reserved.

### 17.2.1.1 TCRBatchMove Class

Transfers records between datasets.

For a list of all members of this type, see [TCRBatchMove](#) members.

### Unit

[CRBatchMove](#)

### Syntax

```
TCRBatchMove = class(TComponent);
```

### Remarks

The TCRBatchMove component transfers records between datasets. Use it to copy dataset records to another dataset or to delete datasets records that match records in another dataset. The [TCRBatchMove.Mode](#) property determines the desired operation type, the [TCRBatchMove.Source](#) and [TCRBatchMove.Destination](#) properties indicate corresponding datasets.

**Note:** A TCRBatchMove component is added to the Data Access page of the component palette, not to the InterBase Access page.

© 1997-2013 Devart. All Rights Reserved.

#### 17.2.1.1.1 Members

[TCRBatchMove](#) class overview.

### Properties

Name	Description
<a href="#">AbortOnKeyViol</a>	Used to specify whether the batch operation should be terminated immediately after key or integrity violation.
<a href="#">AbortOnProblem</a>	Used to specify whether the batch operation should be terminated immediately when it is necessary to truncate data to make it fit the specified Destination.

<a href="#">ChangedCount</a>	Used to get the number of records changed in the destination dataset.
<a href="#">CommitCount</a>	Used to set the number of records to be batch moved before commit occurs.
<a href="#">Destination</a>	Used to specify the destination dataset for the batch operation.
<a href="#">FieldMappingMode</a>	Used to specify the way fields of destination and source datasets will be mapped to each other if the <a href="#">TCRBatchMove.Mappings</a> list is empty.
<a href="#">KeyViolCount</a>	Used to get the number of records that could not be moved to or from the destination dataset because of integrity or key violations.
<a href="#">Mappings</a>	Used to set field matching between source and destination datasets for the batch operation.
<a href="#">Mode</a>	Used to set the type of the batch operation that will be executed after calling the <a href="#">TCRBatchMove.Execute</a> method.
<a href="#">MovedCount</a>	Used to get the number of records that were read from the source dataset during the batch operation.
<a href="#">ProblemCount</a>	Used to get the number of records that could not be added to the destination dataset because of the field type mismatch.
<a href="#">RecordCount</a>	Used to indicate the maximum number of records in the source dataset that will be applied to the destination dataset.
<a href="#">Source</a>	Used to specify the source dataset for the batch operation.

## Methods

Name	Description
------	-------------

[Execute](#)

Performs the batch operation.

**Events**

Name	Description
<a href="#">OnBatchMoveProgress</a>	Occurs when providing feedback to the user about the batch operation in progress is needed.

© 1997-2013 Devart. All Rights Reserved.

## 17.2.1.1.2 Properties

Properties of the **TCRBatchMove** class.

For a complete list of the **TCRBatchMove** class members, see the [TCRBatchMove Members](#) topic.

**Public**

Name	Description
<a href="#">ChangedCount</a>	Used to get the number of records changed in the destination dataset.
<a href="#">KeyViolCount</a>	Used to get the number of records that could not be moved to or from the destination dataset because of integrity or key violations.
<a href="#">MovedCount</a>	Used to get the number of records that were read from the source dataset during the batch operation.
<a href="#">ProblemCount</a>	Used to get the number of records that could not be added to the destination dataset because of the field type mismatch.

**Published**

Name	Description
<a href="#">AbortOnKeyViol</a>	Used to specify whether the batch operation should be terminated immediately after key or integrity violation.

[AbortOnProblem](#)

Used to specify whether the batch operation should be terminated immediately when it is necessary to truncate data to make it fit the specified Destination.

[CommitCount](#)

Used to set the number of records to be batch moved before commit occurs.

[Destination](#)

Used to specify the destination dataset for the batch operation.

[FieldMappingMode](#)

Used to specify the way fields of destination and source datasets will be mapped to each other if the [TCRBatchMove.Mappings](#) list is empty.

[Mappings](#)

Used to set field matching between source and destination datasets for the batch operation.

[Mode](#)

Used to set the type of the batch operation that will be executed after calling the [TCRBatchMove.Execute](#) method.

[RecordCount](#)

Used to indicate the maximum number of records in the source dataset that will be applied to the destination dataset.

[Source](#)

Used to specify the source dataset for the batch operation.

**See Also**

- [TCRBatchMove Class](#)
- [TCRBatchMove Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.2.1.1.2.1 AbortOnKeyViol Property

Used to specify whether the batch operation should be terminated immediately after key or integrity violation.

**Class**[TCRBatchMove](#)**Syntax**

```
property AbortOnKeyViol: boolean default True;
```

**Remarks**

Use the AbortOnKeyViol property to specify whether the batch operation is terminated immediately after key or integrity violation.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.2.1.1.2.2 AbortOnProblem Property

Used to specify whether the batch operation should be terminated immediately when it is necessary to truncate data to make it fit the specified Destination.

**Class**

[TCRBatchMove](#)

**Syntax**

```
property AbortOnProblem: boolean default True;
```

**Remarks**

Use the AbortOnProblem property to specify whether the batch operation is terminated immediately when it is necessary to truncate data to make it fit the specified Destination.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.2.1.1.2.3 ChangedCount Property

Used to get the number of records changed in the destination dataset.

**Class**

[TCRBatchMove](#)

**Syntax**

```
property ChangedCount: Longint;
```

**Remarks**

Use the ChangedCount property to get the number of records changed in the destination dataset. It shows the number of records that were updated in the bmUpdate or bmAppendUpdate mode or were deleted in the bmDelete mode.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.2.1.1.2.4 CommitCount Property

Used to set the number of records to be batch moved before commit occurs.

**Class**

[TCRBatchMove](#)

**Syntax**

```
property CommitCount: integer default 0;
```

### Remarks

Use the CommitCount property to set the number of records to be batch moved before the commit occurs. If it is set to 0, the operation will be chunked to the number of records to fit 32 Kb.

© 1997-2013 Devart. All Rights Reserved.

#### 17.2.1.1.2.5 Destination Property

Used to specify the destination dataset for the batch operation.

### Class

[TCRBatchMove](#)

### Syntax

```
property Destination: TDataSet;
```

### Remarks

Specifies the destination dataset for the batch operation.

© 1997-2013 Devart. All Rights Reserved.

#### 17.2.1.1.2.6 FieldMappingMode Property

Used to specify the way fields of destination and source datasets will be mapped to each other if the [Mappings](#) list is empty.

### Class

[TCRBatchMove](#)

### Syntax

```
property FieldMappingMode: TCRFieldMappingMode default  
mmFieldIndex;
```

### Remarks

Specifies in what way fields of destination and source datasets will be mapped to each other if the [Mappings](#) list is empty.

© 1997-2013 Devart. All Rights Reserved.

#### 17.2.1.1.2.7 KeyViolCount Property

Used to get the number of records that could not be moved to or from the destination dataset because of integrity or key violations.

### Class

[TCRBatchMove](#)

### Syntax

```
property KeyViolCount: Longint;
```

### Remarks

Use the KeyViolCount property to get the number of records that could not be replaced, added, deleted from the destination dataset because of integrity or key violations.

If [AbortOnKeyViol](#) is True, then KeyViolCount will never exceed one, because the operation aborts when the integrity or key violation occurs.

### See Also

- [AbortOnKeyViol](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.2.1.1.2.8 Mappings Property

Used to set field matching between source and destination datasets for the batch operation.

### Class

[TCRBatchMove](#)

### Syntax

```
property Mappings: _TStrings;
```

### Remarks

Use the Mappings property to set field matching between the source and destination datasets for the batch operation. By default fields matching is based on their position in the datasets. To map the column ColName in the source dataset to the column with the same name in the destination dataset, use:  
ColName

### Example

To map a column named SourceColName in the source dataset to the column named DestColName in the destination dataset, use:

```
DestColName=SourceColName
```

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.2.1.1.2.9 Mode Property

Used to set the type of the batch operation that will be executed after calling the [Execute](#) method.

### Class

[TCRBatchMove](#)



## Syntax

```
property Mode: TCRBatchMode default bmAppend;
```

## Remarks

Use the Mode property to set the type of the batch operation that will be executed after calling the [Execute](#) method.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.2.1.1.2.10 MovedCount Property

Used to get the number of records that were read from the source dataset during the batch operation.

## Class

[TCRBatchMove](#)

## Syntax

```
property MovedCount: Longint;
```

## Remarks

Use the MovedCount property to get the number of records that were read from the source dataset during the batch operation. This number includes records that caused key or integrity violations or were trimmed.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.2.1.1.2.11 ProblemCount Property

Used to get the number of records that could not be added to the destination dataset because of the field type mismatch.

## Class

[TCRBatchMove](#)

## Syntax

```
property ProblemCount: Longint;
```

## Remarks

Use the ProblemCount property to get the number of records that could not be added to the destination dataset because of the field type mismatch.

If [AbortOnProblem](#) is True, then ProblemCount will never exceed one, because the operation aborts when the problem occurs.

## See Also

- [AbortOnProblem](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.2.1.1.2.12 RecordCount Property

Used to indicate the maximum number of records in the source dataset that will be applied to the destination dataset.

**Class**

[TCRBatchMove](#)

**Syntax**

```
property RecordCount: Longint default 0;
```

**Remarks**

Determines the maximum number of records in the source dataset, that will be applied to the destination dataset. If it is set to 0, all records in the source dataset will be applied to the destination dataset, starting from the first record. If RecordCount is greater than 0, up to the RecordCount records are applied to the destination dataset, starting from the current record in the source dataset. If RecordCount exceeds the number of records left in the source dataset, batch operation terminates after reaching last record.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.2.1.1.2.13 Source Property

Used to specify the source dataset for the batch operation.

**Class**

[TCRBatchMove](#)

**Syntax**

```
property Source: TDataSet;
```

**Remarks**

Specifies the source dataset for the batch operation.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.2.1.1.3 Methods

Methods of the **TCRBatchMove** class.

For a complete list of the **TCRBatchMove** class members, see the [TCRBatchMove Members](#) topic.

**Public**

Name	Description
<a href="#">Execute</a>	Performs the batch operation.

**See Also**

- [TCRBatchMove Class](#)
  - [TCRBatchMove Class Members](#)
-

© 1997-2013 Devart. All Rights Reserved.

#### 17.2.1.1.3.1 Execute Method

Performs the batch operation.

### Class

[TCRBatchMove](#)

### Syntax

```
procedure Execute;
```

### Remarks

Call the Execute method to perform the batch operation.

© 1997-2013 Devart. All Rights Reserved.

#### 17.2.1.1.4 Events

Events of the **TCRBatchMove** class.

For a complete list of the **TCRBatchMove** class members, see the [TCRBatchMove Members](#) topic.

### Published

Name	Description
<a href="#">OnBatchMoveProgress</a>	Occurs when providing feedback to the user about the batch operation in progress is needed.

### See Also

- [TCRBatchMove Class](#)
- [TCRBatchMove Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.2.1.1.4.1 OnBatchMoveProgress Event

Occurs when providing feedback to the user about the batch operation in progress is needed.

### Class

[TCRBatchMove](#)

### Syntax

```
property OnBatchMoveProgress: TCRBatchMoveProgressEvent;
```

### Remarks

Write the OnBatchMoveProgress event handler to provide feedback to the user about the batch operation progress.

© 1997-2013 Devart. All Rights Reserved.

## 17.2.2 Types

Types in the **CRBatchMove** unit.

### Types

Name	Description
<a href="#">TCRBatchMoveProgressEvent</a>	This type is used for the <a href="#">TCRBatchMove.OnBatchMoveProgress</a> event.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.2.2.1 TCRBatchMoveProgressEvent Procedure Reference

This type is used for the [TCRBatchMove.OnBatchMoveProgress](#) event.

### Unit

[CRBatchMove](#)

### Syntax

```
TCRBatchMoveProgressEvent = procedure (Sender: TObject; Percent:  
integer) of object;
```

### Parameters

*Sender*

An object that raised the event.

*Percent*

Percentage of the batch operation progress.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.2.3 Enumerations

Enumerations in the **CRBatchMove** unit.

### Enumerations

Name	Description
<a href="#">TCRBatchMode</a>	Used to set the type of the batch operation that will be executed after calling the <a href="#">TCRBatchMove.Execute</a> method.
<a href="#">TCRFieldMappingMode</a>	Used to specify the way fields of the destination and source datasets will be mapped to each other if the <a href="#">TCRBatchMove.Mappings</a> list is empty.

---

---

© 1997-2013 Devart. All Rights Reserved.

### 17.2.3.1 TCRBatchMode Enumeration

Used to set the type of the batch operation that will be executed after calling the [TCRBatchMove.Execute](#) method.

#### Unit

[CRBatchMove](#)

#### Syntax

```
TCRBatchMode = (bmAppend, bmUpdate, bmAppendUpdate, bmDelete);
```

#### Values

Value	Meaning
<b>bmAppend</b>	Appends the records from the source dataset to the destination dataset. The default mode.
<b>bmAppendUpdate</b>	Replaces records in the destination dataset with the matching records from the source dataset. If there is no matching record in the destination dataset, the record will be appended to it.
<b>bmDelete</b>	Deletes records from the destination dataset if there are matching records in the source dataset.
<b>bmUpdate</b>	Replaces records in the destination dataset with the matching records from the source dataset.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.2.3.2 TCRFieldMappingMode Enumeration

Used to specify the way fields of the destination and source datasets will be mapped to each other if the [TCRBatchMove.Mappings](#) list is empty.

#### Unit

[CRBatchMove](#)

#### Syntax

```
TCRFieldMappingMode = (mmFieldIndex, mmFieldName);
```

#### Values

Value	Meaning
<b>mmFieldIndex</b>	Specifies that the fields of the destination dataset will be mapped to the fields of the source dataset by field index.
<b>mmFieldName</b>	Mapping is performed by field names.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.3 CRDataTypeMap

This unit contains base classes for Data Type Mapping

### Classes

Name	Description
<a href="#">EDataMappingError</a>	Occurs when unable to map data to a specified type.
<a href="#">EDataTypeMappingError</a>	Base class for errors occurring at data mapping
<a href="#">EInvalidDBTypeMapping</a>	Occurs when DB field type is set incorrectly or when attempting to set Length or Scale for a type that doesn't have such properties.
<a href="#">EInvalidFieldTypeMapping</a>	Occurs when Delphi field type is set incorrectly or when attempting to set Length or Scale for a type that doesn't have such properties.
<a href="#">EUnsupportedDataTypeMapping</a>	Occurs when attempting to register or perform unsupported data type mapping.
<a href="#">TMapRule</a>	Setting rule for data type mapping

© 1997-2013 Devart. All Rights Reserved.

### 17.3.1 Classes

Classes in the **CRDataTypeMap** unit.

### Classes

Name	Description
<a href="#">EDataMappingError</a>	Occurs when unable to map data to a specified type.
<a href="#">EDataTypeMappingError</a>	Base class for errors occurring at data mapping
<a href="#">EInvalidDBTypeMapping</a>	Occurs when DB field type is set incorrectly or when attempting to set Length or Scale for a type that doesn't have such properties.

[EInvalidFieldTypeMapping](#)

Occurs when Delphi field type is set incorrectly or when attempting to set Length or Scale for a type that doesn't have such properties.

[EUnsupportedDataTypeMapping](#)

Occurs when attempting to register or perform unsupported data type mapping.

[TMapRule](#)

Setting rule for data type mapping

© 1997-2013 Devart. All Rights Reserved.

**17.3.1.1 EDataMappingError Class**

Occurs when unable to map data to a specified type.

For a list of all members of this type, see [EDataMappingError](#) members.

**Unit**

[CRDataTypeMap](#)

**Syntax**

```
EDataMappingError = class (EDataTypeMappingError);
```

**Remarks**

EDataMappingError occurs when unable to map data to a specified type. Use EDataMappingError in an exception handling block.

**Inheritance Hierarchy**

[EDataTypeMappingError](#)

**EDataMappingError**

© 1997-2013 Devart. All Rights Reserved.

**17.3.1.1.1 Members**

[EDataMappingError](#) class overview.

© 1997-2013 Devart. All Rights Reserved.

**17.3.1.2 EDataTypeMappingError Class**

Base class for errors occurring at data mapping

For a list of all members of this type, see [EDataTypeMappingError](#) members.

**Unit**

[CRDataTypeMap](#)

**Syntax**

```
EDataTypeMappingError = class (Exception);
```

**Remarks**

Base class for errors occurring at data mapping

---

© 1997-2013 Devart. All Rights Reserved.

## 17.3.1.2.1 Members

[EDataTypeMappingError](#) class overview.

---

© 1997-2013 Devart. All Rights Reserved.

**17.3.1.3 EInvalidDBTypeMapping Class**

Occurs when DB field type is set incorrectly or when attempting to set Length or Scale for a type that doesn't have such properties.

For a list of all members of this type, see [EInvalidDBTypeMapping](#) members.

**Unit**

[CRDataTypeMap](#)

**Syntax**

```
EInvalidDBTypeMapping = class (EDataTypeMappingError) ;
```

**Remarks**

EInvalidDBTypeMapping occurs when DB field type is set incorrectly or when attempting to set Length or Scale for a type that doesn't have such properties. Use EInvalidDBTypeMapping in an exception handling block.

**Inheritance Hierarchy**

[EDataTypeMappingError](#)

**EInvalidDBTypeMapping**

---

© 1997-2013 Devart. All Rights Reserved.

## 17.3.1.3.1 Members

[EInvalidDBTypeMapping](#) class overview.

---

© 1997-2013 Devart. All Rights Reserved.

**17.3.1.4 EInvalidFieldTypeMapping Class**

Occurs when Delphi field type is set incorrectly or when attempting to set Length or Scale for a type that doesn't have such properties.

For a list of all members of this type, see [EInvalidFieldTypeMapping](#) members.

**Unit**

[CRDataTypeMap](#)

**Syntax**

```
EInvalidFieldTypeMapping = class (EDataTypeMappingError) ;
```



## Remarks

EInvalidFieldTypeMapping occurs when Delphi field type is set incorrectly or when attempting to set Length or Scale for a type that doesn't have such properties. Use EInvalidFieldTypeMapping in an exception handling block.

## Inheritance Hierarchy

[EDataTypeMappingError](#)

**EInvalidFieldTypeMapping**

---

© 1997-2013 Devart. All Rights Reserved.

### 17.3.1.4.1 Members

[EInvalidFieldTypeMapping](#) class overview.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.3.1.5 EUnsupportedDataTypeMapping Class

Occurs when attempting to register or perform unsupported data type mapping. For a list of all members of this type, see [EUnsupportedDataTypeMapping](#) members.

## Unit

[CRDataTypeMap](#)

## Syntax

```
EUnsupportedDataTypeMapping = class (EDataTypeMappingError) ;
```

## Remarks

EUnsupportedDataTypeMapping occurs when attempting to register or perform unsupported data type mapping. Use EUnsupportedDataTypeMapping in an exception handling block.

## Inheritance Hierarchy

[EDataTypeMappingError](#)

**EUnsupportedDataTypeMapping**

---

© 1997-2013 Devart. All Rights Reserved.

### 17.3.1.5.1 Members

[EUnsupportedDataTypeMapping](#) class overview.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.3.1.6 TMapRule Class

Setting rule for data type mapping  
For a list of all members of this type, see [TMapRule](#) members.

## Unit

[CRDataTypeMap](#)

## Syntax

```
TMapRule = class (TCollectionItem);
```

© 1997-2013 Devart. All Rights Reserved.

### 17.3.1.6.1 Members

[TMapRule](#) class overview.

## Properties

Name	Description
<a href="#">DBLengthMax</a>	Maximum DB field size
<a href="#">DBLengthMin</a>	Minimum DB field size
<a href="#">DBScaleMax</a>	Maximum DB field scale
<a href="#">DBScaleMin</a>	Minimal DB field scale
<a href="#">DBType</a>	DB field type, that the rule is applied to.
<a href="#">FieldLength</a>	Delphi field length
<a href="#">FieldName</a>	field name in DataSet
<a href="#">FieldScale</a>	Delphi field scale
<a href="#">IgnoreErrors</a>	Ignore data conversion errors. Default value is False.

© 1997-2013 Devart. All Rights Reserved.

### 17.3.1.6.2 Properties

Properties of the **TMapRule** class.

For a complete list of the **TMapRule** class members, see the [TMapRule Members](#) topic.

## Public

Name	Description
<a href="#">DBLengthMax</a>	Maximum DB field size
<a href="#">DBLengthMin</a>	Minimum DB field size
<a href="#">DBScaleMax</a>	Maximum DB field scale
<a href="#">DBScaleMin</a>	Minimal DB field scale
<a href="#">DBType</a>	DB field type, that the rule is applied to.
<a href="#">FieldLength</a>	Delphi field length
<a href="#">FieldName</a>	field name in DataSet
<a href="#">FieldScale</a>	Delphi field scale
<a href="#">IgnoreErrors</a>	Ignore data conversion errors. Default value is False.

## See Also

- [TMapRule Class](#)
- [TMapRule Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.3.1.6.2.1 DBLengthMax Property

Maximum DB field size

##### **Class**

[TMapRule](#)

##### **Syntax**

```
property DBLengthMax: Integer;
```

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.3.1.6.2.2 DBLengthMin Property

Minimum DB field size

##### **Class**

[TMapRule](#)

##### **Syntax**

```
property DBLengthMin: Integer;
```

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.3.1.6.2.3 DBScaleMax Property

Maximum DB field scale

##### **Class**

[TMapRule](#)

##### **Syntax**

```
property DBScaleMax: Integer;
```

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.3.1.6.2.4 DBScaleMin Property

Minimal DB field scale

##### **Class**

[TMapRule](#)

##### **Syntax**

```
property DBScaleMin: Integer;
```

---

© 1997-2013 Devart. All Rights Reserved.

## 17.3.1.6.2.5 DBType Property

DB field type, that the rule is applied to.

**Class**

[TMapRule](#)

**Syntax**

```
property DBType: Word;
```

**Remarks**

Setting DB field type, that the rule is applied to. If the current rule is set for Connection, the rule will be applied to all fields of the specified type in all DataSets related to this Connection.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.3.1.6.2.6 FieldLength Property

Delphi field length

**Class**

[TMapRule](#)

**Syntax**

```
property FieldLength: Integer;
```

---

© 1997-2013 Devart. All Rights Reserved.

## 17.3.1.6.2.7 FieldName Property

field name in DataSet

**Class**

[TMapRule](#)

**Syntax**

```
property FieldName: string;
```

---

© 1997-2013 Devart. All Rights Reserved.

## 17.3.1.6.2.8 FieldScale Property

Delphi field scale

**Class**

[TMapRule](#)

**Syntax**

```
property FieldScale: Integer;
```

---

© 1997-2013 Devart. All Rights Reserved.

## 17.3.1.6.2.9 IgnoreErrors Property

Ignore data conversion errors. Default value is False.

**Class**

[TMapRule](#)

**Syntax**

```
property IgnoreErrors: Boolean;
```

© 1997-2013 Devart. All Rights Reserved.

## 17.4 CREncryption

This unit contains base classes for data encryption.

**Classes****Name**

[TCREncryptor](#)

**Description**

The class that performs data encryption and decryption in a client application using various [encryption algorithms](#).

**Enumerations****Name**

[TCREncDataHeader](#)

**Description**

Specifies whether the additional information is stored with the encrypted data.

[TCREncryptionAlgorithm](#)

Specifies the algorithm of data encryption.

[TCRHashAlgorithm](#)

Specifies the algorithm of generating hash data.

[TCRInvalidHashAction](#)

Specifies the action to perform on data fetching when hash data is invalid.

© 1997-2013 Devart. All Rights Reserved.

### 17.4.1 Classes

Classes in the **CREncryption** unit.

**Classes****Name****Description**

[TCREncryptor](#)

The class that performs data encryption and decryption in a client application using various [encryption algorithms](#).

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.4.1.1 TCREncryptor Class

The class that performs data encryption and decryption in a client application using various [encryption algorithms](#).

For a list of all members of this type, see [TCREncryptor](#) members.

#### Unit

[CREncryption](#)

#### Syntax

```
TCREncryptor = class (TComponent) ;
```

---

© 1997-2013 Devart. All Rights Reserved.

##### 17.4.1.1.1 Members

[TCREncryptor](#) class overview.

#### Properties

Name	Description
<a href="#">DataHeader</a>	Specifies whether the additional information is stored with the encrypted data.
<a href="#">EncryptionAlgorithm</a>	Specifies the algorithm of data encryption.
<a href="#">HashAlgorithm</a>	Specifies the algorithm of generating hash data.
<a href="#">InvalidHashAction</a>	Specifies the action to perform on data fetching when hash data is invalid.
<a href="#">Password</a>	Used to set a password that is used to generate a key for encryption.

#### Methods

Name	Description
<a href="#">SetKey</a>	Sets a key, using which data is encrypted.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.4.1.1.2 Properties

Properties of the **TCREncryptor** class.

For a complete list of the **TCREncryptor** class members, see the [TCREncryptor Members](#) topic.

**Published**

Name	Description
<a href="#">DataHeader</a>	Specifies whether the additional information is stored with the encrypted data.
<a href="#">EncryptionAlgorithm</a>	Specifies the algorithm of data encryption.
<a href="#">HashAlgorithm</a>	Specifies the algorithm of generating hash data.
<a href="#">InvalidHashAction</a>	Specifies the action to perform on data fetching when hash data is invalid.
<a href="#">Password</a>	Used to set a password that is used to generate a key for encryption.

**See Also**

- [TCREncryptor Class](#)
- [TCREncryptor Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.4.1.1.2.1 DataHeader Property

Specifies whether the additional information is stored with the encrypted data.

**Class**

[TCREncryptor](#)

**Syntax**

```
property DataHeader: TCREncDataHeader default ehTagAndHash;
```

**Remarks**

Use DataHeader to specify whether the additional information is stored with the encrypted data. Default value is [ehTagAndHash](#).

---

© 1997-2013 Devart. All Rights Reserved.

## 17.4.1.1.2.2 EncryptionAlgorithm Property

Specifies the algorithm of data encryption.

**Class**

[TCREncryptor](#)

### Syntax

```
property EncryptionAlgorithm: TCREncryptionAlgorithm default eaBlowfish;
```

### Remarks

Use EncryptionAlgorithm to specify the algorithm of data encryption. Default value is [eaBlowfish](#).

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.4.1.1.2.3 HashAlgorithm Property

Specifies the algorithm of generating hash data.

### Class

[TCREncryptor](#)

### Syntax

```
property HashAlgorithm: TCRHashAlgorithm default haSHA1;
```

### Remarks

Use HashAlgorithm to specify the algorithm of generating hash data. This property is used only if hash is stored with the encrypted data (the [DataHeader](#) property is set to [ehTagAndHash](#)). Default value is [haSHA1](#).

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.4.1.1.2.4 InvalidHashAction Property

Specifies the action to perform on data fetching when hash data is invalid.

### Class

[TCREncryptor](#)

### Syntax

```
property InvalidHashAction: TCRInvalidHashAction default ihFail;
```

### Remarks

Use InvalidHashAction to specify the action to perform on data fetching when hash data is invalid. This property is used only if hash is stored with the encrypted data (the [DataHeader](#) property is set to [ehTagAndHash](#)). Default value is [ihFail](#). If the DataHeader property is set to [ehTagAndHash](#), then on data fetching from a server the hash check is performed for each record. After data decryption its hash is calculated and compared with the hash stored in the field. If these values don't coincide, it means that the stored data is incorrect, and depending on the value of the InvalidHashAction property one of the following actions is performed: [ihFail](#) - the EInvalidHash exception is raised and further data reading from the server is interrupted. [ihSkipData](#) - the value of the field for this record is set to Null. No exception is raised. [ihIgnoreError](#) - in spite of the fact that the data is not valid, the value is set in the



field. No exception is raised.

© 1997-2013 Devart. All Rights Reserved.

#### 17.4.1.1.2.5 Password Property

Used to set a password that is used to generate a key for encryption.

### Class

[TCREncryptor](#)

### Syntax

```
property Password: string;
```

### Remarks

Use Password to set a password that is used to generate a key for encryption.

**Note:** Calling of the [SetKey](#) method clears the Password property.

© 1997-2013 Devart. All Rights Reserved.

#### 17.4.1.1.3 Methods

Methods of the **TCREncryptor** class.

For a complete list of the **TCREncryptor** class members, see the [TCREncryptor Members](#) topic.

### Public

Name	Description
<a href="#">SetKey</a>	Sets a key, using which data is encrypted.

### See Also

- [TCREncryptor Class](#)
- [TCREncryptor Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.4.1.1.3.1 SetKey Method

Sets a key, using which data is encrypted.

### Class

[TCREncryptor](#)

### Syntax

```
procedure SetKey(const Key; Count: Integer); overload; procedure
SetKey(const Key: TBytes; Offset: Integer; Count: Integer);
overload;
```

### Parameters

*Key*

Holds bytes that represent a key.

*Offset*

Offset in bytes to the position, where the key begins.

*Count*

Number of bytes to use from Key.

**Remarks**

Use SetKey to set a key, using which data is encrypted.

**Note:** Calling of the SetKey method clears the Password property.

© 1997-2013 Devart. All Rights Reserved.

**17.4.2 Enumerations**

Enumerations in the **CREncryption** unit.

**Enumerations**

Name	Description
<a href="#">TCREncDataHeader</a>	Specifies whether the additional information is stored with the encrypted data.
<a href="#">TCREncryptionAlgorithm</a>	Specifies the algorithm of data encryption.
<a href="#">TCRHashAlgorithm</a>	Specifies the algorithm of generating hash data.
<a href="#">TCRInvalidHashAction</a>	Specifies the action to perform on data fetching when hash data is invalid.

© 1997-2013 Devart. All Rights Reserved.

**17.4.2.1 TCREncDataHeader Enumeration**

Specifies whether the additional information is stored with the encrypted data.

**Unit**

[CREncryption](#)

**Syntax**

```
TCREncDataHeader = (ehTagAndHash, ehTag, ehNone);
```

**Values**

Value	Meaning
<b>ehNone</b>	No additional information is stored.
<b>ehTag</b>	GUID and the random initialization vector are stored with the encrypted data.
<b>ehTagAndHash</b>	Hash, GUID, and the random initialization vector are stored with the encrypted data.

© 1997-2013 Devart. All Rights Reserved.

#### 17.4.2.2 TCREncryptionAlgorithm Enumeration

Specifies the algorithm of data encryption.

##### Unit

[CREncryption](#)

##### Syntax

```
TCREncryptionAlgorithm = (eaTripleDES, eaBlowfish, eaAES128,
    eaAES192, eaAES256, eaCast128, eaRC4);
```

##### Values

Value	Meaning
<b>eaAES128</b>	The AES encryption algorithm with key size of 128 bits is used.
<b>eaAES192</b>	The AES encryption algorithm with key size of 192 bits is used.
<b>eaAES256</b>	The AES encryption algorithm with key size of 256 bits is used.
<b>eaBlowfish</b>	The Blowfish encryption algorithm is used.
<b>eaCast128</b>	The CAST-128 encryption algorithm with key size of 128 bits is used.
<b>eaRC4</b>	The RC4 encryption algorithm is used.
<b>eaTripleDES</b>	The Triple DES encryption algorithm is used.

© 1997-2013 Devart. All Rights Reserved.

#### 17.4.2.3 TCRHashAlgorithm Enumeration

Specifies the algorithm of generating hash data.

##### Unit

[CREncryption](#)

##### Syntax

```
TCRHashAlgorithm = (haSHA1, haMD5);
```

##### Values

Value	Meaning
<b>haMD5</b>	The MD5 hash algorithm is used.
<b>haSHA1</b>	The SHA-1 hash algorithm is used.

© 1997-2013 Devart. All Rights Reserved.

#### 17.4.2.4 TCRInvalidHashAction Enumeration

Specifies the action to perform on data fetching when hash data is invalid.

##### Unit

[CREncryption](#)

##### Syntax

```
TCRInvalidHashAction = (ihFail, ihSkipData, ihIgnoreError);
```

##### Values

Value	Meaning
<b>ihFail</b>	The EInvalidHash exception is raised and further data reading from the server is interrupted.
<b>ihIgnoreError</b>	In spite of the fact that the data is not valid, the value is set in the field. No exception is raised.
<b>ihSkipData</b>	The value of the field for this record is set to Null. No exception is raised.

© 1997-2013 Devart. All Rights Reserved.

## 17.5 DAAlerter

This unit contains the base class for the TIBCAlerter component.

##### Classes

Name	Description
<a href="#">TDAAlerter</a>	A base class that defines functionality for database event notification.

##### Types

Name	Description
<a href="#">TAlertErrorEvent</a>	This type is used for the TDAAlerter.OnError event.

© 1997-2013 Devart. All Rights Reserved.

### 17.5.1 Classes

Classes in the **DAAlerter** unit.

##### Classes

Name	Description
<a href="#">TDAAlerter</a>	A base class that defines functionality for database event notification.

© 1997-2013 Devart. All Rights Reserved.

### 17.5.1.1 TDAAlerter Class

A base class that defines functionality for database event notification. For a list of all members of this type, see [TDAAlerter](#) members.

#### Unit

[DAAlerter](#)

#### Syntax

```
TDAAlerter = class (TComponent) ;
```

#### Remarks

TDAAlerter is a base class that defines functionality for descendant classes support database event notification. Applications never use TDAAlerter objects directly. Instead they use descendants of TDAAlerter.

The TDAAlerter component allows you to register interest in and handle events posted by a database server. Use TDAAlerter to handle events for responding to actions and database changes made by other applications. To get events, an application must register required events. To do this, set the Events property to the required events and call the Start method. When one of the registered events occurs OnEvent handler is called.

© 1997-2013 Devart. All Rights Reserved.

#### 17.5.1.1.1 Members

[TDAAlerter](#) class overview.

#### Properties

Name	Description
<a href="#">Active</a>	Used to determine if TDAAlerter waits for messages.
<a href="#">AutoRegister</a>	Used to automatically register events whenever connection opens.
<a href="#">Connection</a>	Used to specify the connection for TDAAlerter.

#### Methods

Name	Description
<a href="#">SendEvent</a>	Sends an event with Name and content Message.
<a href="#">Start</a>	Starts waiting process.
<a href="#">Stop</a>	Stops waiting process.

#### Events

Name	Description
<a href="#">OnError</a>	Occurs if an exception occurs in waiting process

© 1997-2013 Devart. All Rights Reserved.

#### 17.5.1.1.2 Properties

Properties of the **TDAAlerter** class.

For a complete list of the **TDAAlerter** class members, see the [TDAAlerter Members](#) topic.

#### Public

Name	Description
<a href="#">Active</a>	Used to determine if TDAAlerter waits for messages.
<a href="#">AutoRegister</a>	Used to automatically register events whenever connection opens.
<a href="#">Connection</a>	Used to specify the connection for TDAAlerter.

#### See Also

- [TDAAlerter Class](#)
  - [TDAAlerter Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.5.1.1.2.1 Active Property

Used to determine if TDAAlerter waits for messages.

#### Class

[TDAAlerter](#)

#### Syntax

```
property Active: boolean default False;
```

#### Remarks

Check the Active property to know whether TDAAlerter waits for messages or not. Set it to True to register events.

#### See Also

- [Start](#)
  - [Stop](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.5.1.1.2.2 AutoRegister Property

Used to automatically register events whenever connection opens.

**Class**

[TDAAlerter](#)

**Syntax**

```
property AutoRegister: boolean default False;
```

**Remarks**

Set the AutoRegister property to True to automatically register events whenever connection opens.

© 1997-2013 Devart. All Rights Reserved.

## 17.5.1.1.2.3 Connection Property

Used to specify the connection for TDAAlerter.

**Class**

[TDAAlerter](#)

**Syntax**

```
property Connection: TCustomDAConnection;
```

**Remarks**

Use the Connection property to specify the connection for TDAAlerter.

**See Also**

- [TIBCCConnection](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.5.1.1.3 Methods

Methods of the **TDAAlerter** class.

For a complete list of the **TDAAlerter** class members, see the [TDAAlerter Members](#) topic.

**Public**

Name	Description
<a href="#">SendEvent</a>	Sends an event with Name and content Message.
<a href="#">Start</a>	Starts waiting process.
<a href="#">Stop</a>	Stops waiting process.

**See Also**

- [TDAAlerter Class](#)

- [TDAAlerter Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.5.1.1.3.1 SendEvent Method

Sends an event with Name and content Message.

#### Class

[TDAAlerter](#)

#### Syntax

```
procedure SendEvent(const EventName: string; const Message:  
string);
```

#### Parameters

*EventName*

Holds the event name.

*Message*

Holds the content Message of the event.

#### Remarks

Use SendEvent procedure to send an event with Name and content Message.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.5.1.1.3.2 Start Method

Starts waiting process.

#### Class

[TDAAlerter](#)

#### Syntax

```
procedure Start;
```

#### Remarks

Call the Start method to run waiting process. After starting TDAAlerter waits for messages with names defined by the Events property.

#### See Also

- [Stop](#)
  - [Active](#)
  - [TIBCAlerter.OnEvent](#)
- 

© 1997-2013 Devart. All Rights Reserved.



## 17.5.1.1.3.3 Stop Method

Stops waiting process.

**Class**

[TDAAlerter](#)

**Syntax**

```
procedure Stop;
```

**Remarks**

Call Stop method to end waiting process.

**See Also**

- [Start](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.5.1.1.4 Events

Events of the **TDAAlerter** class.

For a complete list of the **TDAAlerter** class members, see the [TDAAlerter Members](#) topic.

**Public**

Name	Description
<a href="#">OnError</a>	Occurs if an exception occurs in waiting process

**See Also**

- [TDAAlerter Class](#)
- [TDAAlerter Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.5.1.1.4.1 OnError Event

Occurs if an exception occurs in waiting process

**Class**

[TDAAlerter](#)

**Syntax**

```
property OnError: TAlerterErrorEvent;
```

**Remarks**

The OnError event occurs if an exception occurs in waiting process. Alerter stops in this case. The exception can be accessed using the E parameter.

© 1997-2013 Devart. All Rights Reserved.

## 17.5.2 Types

Types in the **DAAlerter** unit.

### Types

Name	Description
<a href="#">TAlerterErrorEvent</a>	This type is used for the TDAAlerter.OnError event.

© 1997-2013 Devart. All Rights Reserved.

### 17.5.2.1 TAlerterErrorEvent Procedure Reference

This type is used for the TDAAlerter.OnError event.

### Unit

[DAAlerter](#)

### Syntax

```
TAlerterErrorEvent = procedure (Sender: TDAAlerter; E: Exception)
of object;
```

### Parameters

*Sender*

An object that raised the event.

*E*

Exception object.

© 1997-2013 Devart. All Rights Reserved.

## 17.6 DADump

This unit contains the base class for the TIBCDump component.

### Classes

Name	Description
<a href="#">TDADump</a>	A base class that defines functionality for descendant classes that dump database objects to a script.
<a href="#">TDADumpOptions</a>	This class allows setting up the behaviour of the TDADump class.

### Types

Name	Description
------	-------------

[TDABackupProgressEvent](#)

This type is used for the [TDADump.OnBackupProgress](#) event.

[TDARestoreProgressEvent](#)

This type is used for the [TDADump.OnRestoreProgress](#) event.

© 1997-2013 Devart. All Rights Reserved.

## 17.6.1 Classes

Classes in the **DADump** unit.

### Classes

Name	Description
<a href="#">TDADump</a>	A base class that defines functionality for descendant classes that dump database objects to a script.
<a href="#">TDADumpOptions</a>	This class allows setting up the behaviour of the TDADump class.

© 1997-2013 Devart. All Rights Reserved.

### 17.6.1.1 TDADump Class

A base class that defines functionality for descendant classes that dump database objects to a script.

For a list of all members of this type, see [TDADump](#) members.

### Unit

[DADump](#)

### Syntax

```
TDADump = class (TComponent) ;
```

### Remarks

TDADump is a base class that defines functionality for descendant classes that dump database objects to a script. Applications never use TDADump objects directly. Instead they use descendants of TDADump.

Use TDADump descendants to dump database objects, such as tables, stored procedures, and functions for backup or for transferring the data to another SQL server. The dump contains SQL statements to create the table or other database objects and/or populate the table.

© 1997-2013 Devart. All Rights Reserved.

## 17.6.1.1.1 Members

[TDADump](#) class overview.

## Properties

Name	Description
<a href="#">Connection</a>	Used to specify a connection object that will be used to connect to a data store.
<a href="#">Debug</a>	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#">Options</a>	Used to specify the behaviour of a TDADump component.
<a href="#">SQL</a>	Used to set or get the dump script.
<a href="#">TableNames</a>	Used to set the names of the tables to dump.

## Methods

Name	Description
<a href="#">Backup</a>	Dumps database objects to the <a href="#">TDADump.SQL</a> property.
<a href="#">BackupQuery</a>	Dumps the results of a particular query.
<a href="#">BackupToFile</a>	Dumps database objects to the specified file.
<a href="#">BackupToStream</a>	Dumps database objects to the stream.
<a href="#">Restore</a>	Executes a script contained in the SQL property.
<a href="#">RestoreFromFile</a>	Executes a script from a file.
<a href="#">RestoreFromStream</a>	Executes a script received from the stream.

## Events

Name	Description
<a href="#">OnBackupProgress</a>	Occurs to indicate the <a href="#">TDADump.Backup</a> , M: Devart.Dac.TDADump.BackupToFile(System.String) or M: Devart.Dac.TDADump.BackupToStream (Borland.Vcl.TStream) method execution progress.

[OnError](#)

Occurs when InterBase raises some error on [TDADump.Restore](#).

[OnRestoreProgress](#)

Occurs to indicate the [TDADump.Restore](#), [TDADump.RestoreFromFile](#), or [TDADump.RestoreFromStream](#) method execution progress.

© 1997-2013 Devart. All Rights Reserved.

## 17.6.1.1.2 Properties

Properties of the **TDADump** class.

For a complete list of the **TDADump** class members, see the [TDADump Members](#) topic.

**Public****Name**[Connection](#)**Description**

Used to specify a connection object that will be used to connect to a data store.

[Options](#)

Used to specify the behaviour of a TDADump component.

**Published****Name**[Debug](#)**Description**

Used to display executing statement, all its parameters' values, and the type of parameters.

[SQL](#)

Used to set or get the dump script.

[TableNames](#)

Used to set the names of the tables to dump.

**See Also**

- [TDADump Class](#)
- [TDADump Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.6.1.1.2.1 Connection Property

Used to specify a connection object that will be used to connect to a data store.

**Class**[TDADump](#)

**Syntax**

**property** Connection: [TCustomDAConnection](#);

**Remarks**

Use the Connection property to specify a connection object that will be used to connect to a data store.

Set at design-time by selecting from the list of provided TCustomDAConnection or its descendant class objects.

At runtime, link an instance of a TCustomDAConnection descendant to the Connection property.

**See Also**

- [TCustomDAConnection](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.6.1.1.2.2 Debug Property

Used to display executing statement, all its parameters' values, and the type of parameters.

**Class**

[TDADump](#)

**Syntax**

**property** Debug: boolean **default** False;

**Remarks**

Used to display executing statement, all its parameters' values, and the type of parameters.

**See Also**

- [TCustomDADataset.Debug](#)
  - [TCustomDASQL.Debug](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.6.1.1.2.3 Options Property

Used to specify the behaviour of a TDADump component.

**Class**

[TDADump](#)

**Syntax**

**property** Options: [TDADumpOptions](#);

**Remarks**

Use the Options property to specify the behaviour of a TDADump component. Descriptions of all options are in the table below.

Option Name	Description
<a href="#">AddDrop</a>	Used to add drop statements to a script before creating statements.
<a href="#">GenerateHeader</a>	Used to add a comment header to a script.
<a href="#">QuoteNames</a>	Used for TDADump to quote all database object names in generated SQL statements.

©

1997-2013 Devart. All Rights Reserved.

## 17.6.1.1.2.4 SQL Property

Used to set or get the dump script.

**Class**

[TDADump](#)

**Syntax**

```
property SQL: _TStrings;
```

**Remarks**

Use the SQL property to get or set the dump script. The SQL property stores script that is executed by the [Restore](#) method. This property will store the result of [Backup](#) and [BackupQuery](#). At design time the SQL property can be edited by invoking the String List editor in Object Inspector.

**See Also**

- [Restore](#)
- [Backup](#)
- [BackupQuery](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.6.1.1.2.5 TableNames Property

Used to set the names of the tables to dump.

**Class**

[TDADump](#)

**Syntax**

```
property TableNames: string;
```

### Remarks

Use the TableNames property to set the names of the tables to dump. Table names must be separated with commas. If it is empty, the [Backup](#) method will dump all available tables.

### See Also

- [Backup](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.6.1.1.3 Methods

Methods of the **TDADump** class.

For a complete list of the **TDADump** class members, see the [TDADump Members](#) topic.

### Public

Name	Description
<a href="#">Backup</a>	Dumps database objects to the <a href="#">TDADump.SQL</a> property.
<a href="#">BackupQuery</a>	Dumps the results of a particular query.
<a href="#">BackupToFile</a>	Dumps database objects to the specified file.
<a href="#">BackupToStream</a>	Dumps database objects to the stream.
<a href="#">Restore</a>	Executes a script contained in the SQL property.
<a href="#">RestoreFromFile</a>	Executes a script from a file.
<a href="#">RestoreFromStream</a>	Executes a script received from the stream.

### See Also

- [TDADump Class](#)
  - [TDADump Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.6.1.1.3.1 Backup Method

Dumps database objects to the [SQL](#) property.

### Class

[TDADump](#)

### Syntax



```
procedure Backup;
```

### Remarks

Call the Backup method to dump database objects. The result script will be stored in the [SQL](#) property.

### See Also

- [SQL](#)
- [Restore](#)
- [BackupToFile](#)
- [BackupToStream](#)
- [BackupQuery](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.6.1.1.3.2 BackupQuery Method

Dumps the results of a particular query.

### Class

[TDADump](#)

### Syntax

```
procedure BackupQuery(const Query: string);
```

#### Parameters

*Query*

Holds a query used for data selection.

### Remarks

Call the BackupQuery method to dump the results of a particular query. Query must be a valid select statement. If this query selects data from several tables, only data of the first table in the from list will be dumped.

### See Also

- [Restore](#)
- [Backup](#)
- [BackupToFile](#)
- [BackupToStream](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.6.1.1.3.3 BackupToFile Method

Dumps database objects to the specified file.

### Class

[TDADump](#)

### Syntax

```
procedure BackupToFile(const FileName: string; const Query: string  
= '');
```

#### Parameters

*FileName*

Holds the file name to dump database objects to.

*Query*

Your query to receive the data for dumping.

### Remarks

Call the BackupToFile method to dump database objects to the specified file.

### See Also

- [RestoreFromStream](#)
  - [Backup](#)
  - [BackupToStream](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.6.1.1.3.4 BackupToStream Method

Dumps database objects to the stream.

### Class

[TDADump](#)

### Syntax

```
procedure BackupToStream(Stream: TStream; const Query: string = '  
' );
```

#### Parameters

*Stream*

Holds the stream to dump database objects to.

*Query*

Your query to receive the data for dumping.

### Remarks

Call the BackupToStream method to dump database objects to the stream.

### See Also

- [RestoreFromStream](#)
  - [Backup](#)
  - [BackupToFile](#)
-

© 1997-2013 Devart. All Rights Reserved.

#### 17.6.1.1.3.5 Restore Method

Executes a script contained in the SQL property.

### Class

[TDADump](#)

### Syntax

```
procedure Restore;
```

### Remarks

Call the Restore method to execute a script contained in the SQL property.

### See Also

- [RestoreFromFile](#)
- [RestoreFromStream](#)
- [Backup](#)
- [SQL](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.6.1.1.3.6 RestoreFromFile Method

Executes a script from a file.

### Class

[TDADump](#)

### Syntax

```
procedure RestoreFromFile(const FileName: string);
```

#### Parameters

*FileName*

Holds the file name to execute a script from.

### Remarks

Call the RestoreFromFile method to execute a script from the specified file.

### See Also

- [Restore](#)
- [RestoreFromStream](#)
- [BackupToFile](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.6.1.1.3.7 RestoreFromStream Method

Executes a script received from the stream.

**Class**

[TDADump](#)

**Syntax**

```
procedure RestoreFromStream(Stream: TStream);
```

**Parameters**

*Stream*

Holds a stream to receive a script to be executed.

**Remarks**

Call the RestoreFromStream method to execute a script received from the stream.

**See Also**

- [Restore](#)
  - [RestoreFromFile](#)
  - [BackupToStream](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.6.1.1.4 Events

Events of the **TDADump** class.

For a complete list of the **TDADump** class members, see the [TDADump Members](#) topic.

**Published**

Name	Description
<a href="#">OnBackupProgress</a>	Occurs to indicate the <a href="#">TDADump.Backup</a> , M: Devart.Dac.TDADump.BackupToFile(System.String) or M: Devart.Dac.TDADump.BackupToStream (Borland.Vcl.TStream) method execution progress.
<a href="#">OnError</a>	Occurs when InterBase raises some error on <a href="#">TDADump.Restore</a> .
<a href="#">OnRestoreProgress</a>	Occurs to indicate the <a href="#">TDADump.Restore</a> , <a href="#">TDADump.RestoreFromFile</a> , or <a href="#">TDADump.RestoreFromStream</a> method execution progress.

## See Also

- [TDADump Class](#)
- [TDADump Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.6.1.1.4.1 OnBackupProgress Event

Occurs to indicate the [Backup](#), M:Devart.Dac.TDADump.BackupToFile(System.String) or M:Devart.Dac.TDADump.BackupToStream(Borland.Vcl.TStream) method execution progress.

## Class

[TDADump](#)

## Syntax

```
property OnBackupProgress: TDABackupProgressEvent;
```

## Remarks

The OnBackupProgress event occurs several times during the dumping process of the [Backup](#), M:Devart.Dac.TDADump.BackupToFile(System.String), or M:Devart.Dac.TDADump.BackupToStream(Borland.Vcl.TStream) method execution and indicates its progress. ObjectName parameter indicates the name of the currently dumping database object. ObjectNum shows the number of the current database object in the backup queue starting from zero. ObjectCount shows the quantity of database objects to dump. Percent parameter shows the current percentage of the current table data dumped, not the current percentage of the entire dump process.

## See Also

- [Backup](#)
- [BackupToFile](#)
- [BackupToStream](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.6.1.1.4.2 OnError Event

Occurs when InterBase raises some error on [Restore](#).

## Class

[TDADump](#)

## Syntax

```
property OnError: TOnErrorEvent;
```

## Remarks

The OnError event occurs when InterBase raises some error on [Restore](#). Action indicates the action to take when the OnError handler exits. On entry into the handler, Action is always set to eaException.

**Note:** You should add the DAScript module to the 'uses' list to use the OnError event handler.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.6.1.1.4.3 OnRestoreProgress Event

Occurs to indicate the [Restore](#), [RestoreFromFile](#), or [RestoreFromStream](#) method execution progress.

### Class

[TDADump](#)

### Syntax

```
property OnRestoreProgress: TDARestoreProgressEvent;
```

### Remarks

The OnRestoreProgress event occurs several times during the dumping process of the [Restore](#), [RestoreFromFile](#), or [RestoreFromStream](#) method execution and indicates its progress. The Percent parameter of the OnRestoreProgress event handler indicates the percentage of the whole restore script execution.

### See Also

- [Restore](#)
  - [RestoreFromFile](#)
  - [RestoreFromStream](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.6.1.2 TDADumpOptions Class

This class allows setting up the behaviour of the TDADump class.  
For a list of all members of this type, see [TDADumpOptions](#) members.

### Unit

[DADump](#)

### Syntax

```
TDADumpOptions = class (TPersistent);
```

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.6.1.2.1 Members

[TDADumpOptions](#) class overview.

### Properties

Name	Description
------	-------------

[AddDrop](#)

Used to add drop statements to a script before creating statements.

[GenerateHeader](#)

Used to add a comment header to a script.

[QuoteNames](#)

Used for TDADump to quote all database object names in generated SQL statements.

© 1997-2013 Devart. All Rights Reserved.

#### 17.6.1.2.2 Properties

Properties of the **TDADumpOptions** class.

For a complete list of the **TDADumpOptions** class members, see the [TDADumpOptions Members](#) topic.

### Published

Name	Description
<a href="#">AddDrop</a>	Used to add drop statements to a script before creating statements.
<a href="#">GenerateHeader</a>	Used to add a comment header to a script.
<a href="#">QuoteNames</a>	Used for TDADump to quote all database object names in generated SQL statements.

### See Also

- [TDADumpOptions Class](#)
- [TDADumpOptions Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.6.1.2.2.1 AddDrop Property

Used to add drop statements to a script before creating statements.

### Class

[TDADumpOptions](#)

### Syntax

```
property AddDrop: boolean default True;
```

### Remarks

Use the AddDrop property to add drop statements to a script before creating statements.

© 1997-2013 Devart. All Rights Reserved.

## 17.6.1.2.2.2 GenerateHeader Property

Used to add a comment header to a script.

**Class**

[TDADumpOptions](#)

**Syntax**

```
property GenerateHeader: boolean default True;
```

**Remarks**

Use the GenerateHeader property to add a comment header to a script. It contains script generation date, DAC version, and some other information.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.6.1.2.2.3 QuoteNames Property

Used for TDADump to quote all database object names in generated SQL statements.

**Class**

[TDADumpOptions](#)

**Syntax**

```
property QuoteNames: boolean default False;
```

**Remarks**

If the QuoteNames property is True, TDADump quotes all database object names in generated SQL statements.

---

© 1997-2013 Devart. All Rights Reserved.

**17.6.2 Types**

Types in the **DADump** unit.

**Types**

Name	Description
<a href="#">TDABackupProgressEvent</a>	This type is used for the <a href="#">TDADump.OnBackupProgress</a> event.
<a href="#">TDARestoreProgressEvent</a>	This type is used for the <a href="#">TDADump.OnRestoreProgress</a> event.

---

© 1997-2013 Devart. All Rights Reserved.



### 17.6.2.1 TDABackupProgressEvent Procedure Reference

This type is used for the [TDADump.OnBackupProgress](#) event.

#### Unit

[DADump](#)

#### Syntax

```
TDABackupProgressEvent = procedure (Sender: TObject; ObjectName: string; ObjectNum: integer; ObjectCount: integer; Percent: integer) of object;
```

#### Parameters

*Sender*

An object that raised the event.

*ObjectName*

The name of the currently dumping database object.

*ObjectNum*

The number of the current database object in the backup queue starting from zero.

*ObjectCount*

The quantity of database objects to dump.

*Percent*

The current percentage of the current table data dumped.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.6.2.2 TDARestoreProgressEvent Procedure Reference

This type is used for the [TDADump.OnRestoreProgress](#) event.

#### Unit

[DADump](#)

#### Syntax

```
TDARestoreProgressEvent = procedure (Sender: TObject; Percent: integer) of object;
```

#### Parameters

*Sender*

An object that raised the event.

*Percent*

The percentage of the whole restore script execution.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.7 DALoader

This unit contains the base class for the TIBCLoader component.

#### Classes

Name	Description
<a href="#">TDAColumn</a>	Represents the attributes for column loading.
<a href="#">TDAColumns</a>	Holds a collection of <a href="#">TDAColumn</a> objects.
<a href="#">TDALoader</a>	This class allows loading external data into database.

### Types

Name	Description
<a href="#">TDAPutDataEvent</a>	This type is used for the <a href="#">TDALoader.OnPutData</a> event.
<a href="#">TGetColumnDataEvent</a>	This type is used for the <a href="#">TDALoader.OnGetColumnData</a> event.
<a href="#">TLoaderProgressEvent</a>	This type is used for the <a href="#">TDALoader.OnProgress</a> event.

© 1997-2013 Devart. All Rights Reserved.

## 17.7.1 Classes

Classes in the **DALoader** unit.

### Classes

Name	Description
<a href="#">TDAColumn</a>	Represents the attributes for column loading.
<a href="#">TDAColumns</a>	Holds a collection of <a href="#">TDAColumn</a> objects.
<a href="#">TDALoader</a>	This class allows loading external data into database.

© 1997-2013 Devart. All Rights Reserved.

### 17.7.1.1 TDAColumn Class

Represents the attributes for column loading.  
For a list of all members of this type, see [TDAColumn](#) members.

### Unit

[DALoader](#)

### Syntax

```
TDAColumn = class (TCollectionItem);
```

### Remarks

Each [TDALoader](#) uses [TDAColumns](#) to maintain a collection of TDAColumn objects. TDAColumn object represents the attributes for column loading. Every TDAColumn object corresponds to one of the table fields with the same name as its [TDAColumn.Name](#) property.

To create columns at design-time use the column editor of the [TDALoader](#) component.

## See Also

- [TDALoader](#)
- [TDAColumns](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.7.1.1.1 Members

[TDAColumn](#) class overview.

## Properties

Name	Description
<a href="#">FieldType</a>	Used to specify the types of values that will be loaded.
<a href="#">Name</a>	Used to specify the field name of loading table.

© 1997-2013 Devart. All Rights Reserved.

### 17.7.1.1.2 Properties

Properties of the **TDAColumn** class.

For a complete list of the **TDAColumn** class members, see the [TDAColumn Members](#) topic.

## Published

Name	Description
<a href="#">FieldType</a>	Used to specify the types of values that will be loaded.
<a href="#">Name</a>	Used to specify the field name of loading table.

## See Also

- [TDAColumn Class](#)
- [TDAColumn Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.7.1.1.2.1 FieldType Property

Used to specify the types of values that will be loaded.

## Class

[TDAColumn](#)

### Syntax

```
property FieldType: TFieldType default ftString;
```

### Remarks

Use the FieldType property to specify the types of values that will be loaded. Field types for columns may not match data types for the corresponding fields in the database table. [TDALoader](#) will cast data values to the types of their fields.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.7.1.1.2.2 Name Property

Used to specify the field name of loading table.

### Class

[TDAColumn](#)

### Syntax

```
property Name: string;
```

### Remarks

Each TDAColumn corresponds to one field of the loading table. Use the Name property to specify the name of this field.

### See Also

- [FieldType](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.7.1.2 TDAColumns Class

Holds a collection of [TDAColumn](#) objects.  
For a list of all members of this type, see [TDAColumns](#) members.

### Unit

[DALoader](#)

### Syntax

```
TDAColumns = class(TOwnedCollection);
```

### Remarks

Each TDAColumns holds a collection of [TDAColumn](#) objects. TDAColumns maintains an index of the columns in its Items array. The Count property contains the number of columns in the collection. At design-time, use the Columns editor to add, remove, or modify columns.

## See Also

- [TDALoader](#)
- [TDAColumn](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.7.1.2.1 Members

[TDAColumns](#) class overview.

## Properties

Name	Description
<a href="#">Items</a>	Used to access individual columns.

© 1997-2013 Devart. All Rights Reserved.

### 17.7.1.2.2 Properties

Properties of the **TDAColumns** class.  
For a complete list of the **TDAColumns** class members, see the [TDAColumns Members](#) topic.

## Public

Name	Description
<a href="#">Items</a>	Used to access individual columns.

## See Also

- [TDAColumns Class](#)
- [TDAColumns Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.7.1.2.2.1 Items Property(Indexer)

Used to access individual columns.

## Class

[TDAColumns](#)

## Syntax

```
property Items[Index: integer]: TDAColumn; default;
```

### Parameters

#### Index

Holds the Index of [TDAColumn](#) to refer to.

## Remarks

Use the Items property to access individual columns. The value of the Index

parameter corresponds to the Index property of [TDAColumn](#).

## See Also

- [TDAColumn](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.7.1.3 TDALoader Class

This class allows loading external data into database.  
For a list of all members of this type, see [TDALoader](#) members.

## Unit

[DALoader](#)

## Syntax

```
TDALoader = class (TComponent);
```

## Remarks

TDALoader allows loading external data into database. To specify the name of loading table set the [TDALoader.TableName](#) property. Use the [TDALoader.Columns](#) property to access individual columns. Write the [TDALoader.OnGetColumnData](#) or [TDALoader.OnPutData](#) event handlers to read external data and pass it to the database. Call the [TDALoader.Load](#) method to start loading data.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.7.1.3.1 Members

[TDALoader](#) class overview.

## Properties

Name	Description
<a href="#">Columns</a>	Used to add a <a href="#">TDAColumn</a> object for each field that will be loaded.
<a href="#">Connection</a>	Used to specify TCustomDAConnection in which TDALoader will be executed.
<a href="#">TableName</a>	Used to specify the name of the table to which data will be loaded.

## Methods

Name	Description
------	-------------

[CreateColumns](#)

Creates [TDAColumn](#) objects for all fields of the table with the same name as [TDALoader.TableName](#).

[Load](#)

Starts loading data.

[LoadFromDataSet](#)

Loads data from the specified dataset.

[PutColumnData](#)

Overloaded. Puts the value of individual columns.

## Events

Name	Description
<a href="#">OnGetColumnData</a>	Occurs when it is needed to put column values.
<a href="#">OnProgress</a>	Occurs if handling data loading progress of the <a href="#">TDALoader</a> . <a href="#">LoadFromDataSet</a> method is needed.
<a href="#">OnPutData</a>	Occurs when putting loading data by rows is needed.

© 1997-2013 Devart. All Rights Reserved.

### 17.7.1.3.2 Properties

Properties of the **TDALoader** class.

For a complete list of the **TDALoader** class members, see the [TDALoader Members](#) topic.

## Public

Name	Description
<a href="#">Columns</a>	Used to add a <a href="#">TDAColumn</a> object for each field that will be loaded.
<a href="#">Connection</a>	Used to specify TCustomDACConnection in which TDALoader will be executed.
<a href="#">TableName</a>	Used to specify the name of the table to which data will be loaded.

## See Also

- [TDALoader Class](#)
- [TDALoader Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.7.1.3.2.1 Columns Property

Used to add a [TDAColumn](#) object for each field that will be loaded.

**Class**

[TDALoader](#)

**Syntax**

```
property Columns: TDAColumns stored IsColumnsStored;
```

**Remarks**

Use the Columns property to add a [TDAColumn](#) object for each field that will be loaded.

**See Also**

- [TDAColumns](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.7.1.3.2.2 Connection Property

Used to specify TCustomDAConnection in which TDALoader will be executed.

**Class**

[TDALoader](#)

**Syntax**

```
property Connection: TCustomDAConnection;
```

**Remarks**

Use the Connection property to specify TCustomDAConnection in which TDALoader will be executed. If Connection is not connected, the [Load](#) method calls [TCustomDAConnection.Connect](#).

**See Also**

- [TCustomDAConnection](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.7.1.3.2.3 TableName Property

Used to specify the name of the table to which data will be loaded.

**Class**

[TDALoader](#)

**Syntax**



```
property TableName: string;
```

### Remarks

Set the TableName property to specify the name of the table to which data will be loaded. Add TDAColumn objects to [Columns](#) for the fields that are needed to be loaded.

### See Also

- [TDAColumn](#)
- [TCustomDAConnection.GetTableNames](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.7.1.3.3 Methods

Methods of the **TDALoader** class.

For a complete list of the **TDALoader** class members, see the [TDALoader Members](#) topic.

### Public

Name	Description
<a href="#">CreateColumns</a>	Creates <a href="#">TDAColumn</a> objects for all fields of the table with the same name as <a href="#">TDALoader.TableName</a> .
<a href="#">Load</a>	Starts loading data.
<a href="#">LoadFromDataSet</a>	Loads data from the specified dataset.
<a href="#">PutColumnData</a>	Overloaded. Puts the value of individual columns.

### See Also

- [TDALoader Class](#)
- [TDALoader Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.7.1.3.3.1 CreateColumns Method

Creates [TDAColumn](#) objects for all fields of the table with the same name as [TableName](#).

### Class

[TDALoader](#)

### Syntax

```
procedure CreateColumns;
```

### Remarks

Call the CreateColumns method to create [TDAColumn](#) objects for all fields of the table with the same name as [TableName](#). If columns were created before, they will be recreated. You can call CreateColumns from the component popup menu at design-time. After you can customize column loading by setting properties of TDAColumn objects.

### See Also

- [TDAColumn](#)
  - [TableName](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.7.1.3.3.2 Load Method

Starts loading data.

### Class

[TDALoader](#)

### Syntax

```
procedure Load; virtual;
```

### Remarks

Call the Load method to start loading data. At first it is necessary to [create columns](#) and write one of the [OnPutData](#) or [OnGetColumnData](#) event handlers.

### See Also

- [OnGetColumnData](#)
  - [OnPutData](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.7.1.3.3.3 LoadFromDataSet Method

Loads data from the specified dataset.

### Class

[TDALoader](#)

### Syntax

```
procedure LoadFromDataSet (DataSet: TDataSet);
```

### Parameters

*DataSet*

Holds the dataset to load data from.

### Remarks

Call the LoadFromDataSet method to load data from the specified dataset. There is no need to create columns and write event handlers for [OnPutData](#) and [OnGetColumnData](#) before calling this method.

© 1997-2013 Devart. All Rights Reserved.

#### 17.7.1.3.3.4 PutColumnData Method

Puts the value of individual columns.

### Class

[TDALoader](#)

### Overload List

Name	Description
<a href="#">PutColumnData(Col: integer; Row: integer; <b>const</b> Value: variant)</a>	Puts the value of individual columns by the column index.
<a href="#">PutColumnData(<b>const</b> ColName: string; Row: integer; <b>const</b> Value: variant)</a>	Puts the value of individual columns by the column name.

© 1997-2013 Devart. All Rights Reserved.

Puts the value of individual columns by the column index.

### Class

[TDALoader](#)

### Syntax

```
procedure PutColumnData(Col: integer; Row: integer; const Value: variant); overload; virtual
```

#### Parameters

*Col*

Holds the index of a loading column. The first column has index 0.

*Row*

Holds the number of loading row. Row starts from 1.

*Value*

Holds the column value.

### Remarks

Call the PutColumnData method to put the value of individual columns. The Col parameter indicates the index of loading column. The first column has index 0. The Row parameter indicates the number of the loading row. Row starts from 1. This overloaded method works faster because it searches the right index by its index, not by the index name. The value of a column should be assigned to the Value parameter.

### See Also

- [TDALoader.OnPutData](#)

© 1997-2013 Devart. All Rights Reserved.

Puts the value of individual columns by the column name.

## Class

[TDALoader](#)

## Syntax

```
procedure PutColumnData(const ColName: string; Row: integer; const
Value: variant); overload
```

### Parameters

*ColName*

Holds the name of a loading column.

*Row*

Holds the number of loading row. Row starts from 1.

*Value*

Holds the column value.

© 1997-2013 Devart. All Rights Reserved.

#### 17.7.1.3.4 Events

Events of the **TDALoader** class.

For a complete list of the **TDALoader** class members, see the [TDALoader Members](#) topic.

## Public

Name	Description
<a href="#">OnGetColumnData</a>	Occurs when it is needed to put column values.
<a href="#">OnProgress</a>	Occurs if handling data loading progress of the <a href="#">TDALoader.LoadFromDataSet</a> method is needed.
<a href="#">OnPutData</a>	Occurs when putting loading data by rows is needed.

## See Also

- [TDALoader Class](#)
- [TDALoader Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.7.1.3.4.1 OnGetColumnData Event

Occurs when it is needed to put column values.

**Class**

[TDALoader](#)

**Syntax**

**property** OnGetColumnData: [TGetColumnDataEvent](#);

**Remarks**

Write the OnGetColumnData event handler to put column values. [TDALoader](#) calls the OnGetColumnData event handler for each column in the loop. Column points to a [TDAColumn](#) object that corresponds to the current loading column. Use its Name or Index property to identify what column is loading. The Row parameter indicates the current loading record. TDALoader increments the Row parameter when all the columns of the current record are loaded. The first row is 1. Set EOF to True to stop data loading. Fill the Value parameter by column values. To start loading call the [Load](#) method.

Another way to load data is using the [OnPutData](#) event.

**Example**

This handler loads 1000 rows.

```
procedure TfmMain.GetColumnData(Sender: TObject;
    Column: TDAColumn; Row: Integer; var Value: Variant;
    var EOF: Boolean);
begin
    if Row <= 1000 then begin
        case Column.Index of
            0: Value := Row;
            1: Value := Random(100);
            2: Value := Random*100;
            3: Value := 'abc01234567890123456789';
            4: Value := Date;
        else
            Value := Null;
        end;
    end
    else
        EOF := True;
    end;
```

**See Also**

- [OnPutData](#)
- [Load](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.7.1.3.4.2 OnProgress Event

Occurs if handling data loading progress of the [LoadFromDataSet](#) method is needed.

**Class**

[TDALoader](#)

**Syntax**

```
property OnProgress: TLoaderProgressEvent;
```

**Remarks**

Add a handler to this event if you want to handle data loading progress of the [LoadFromDataSet](#) method.

**See Also**

- [LoadFromDataSet](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.7.1.3.4.3 OnPutData Event

Occurs when putting loading data by rows is needed.

**Class**

[TDALoader](#)

**Syntax**

```
property OnPutData: TDAPutDataEvent;
```

**Remarks**

Write the OnPutData event handler to put loading data by rows.  
Note that rows should be loaded from the first in the ascending order.  
To start loading, call the [Load](#) method. It is more effective way to load data in comparison with using [OnGetColumnData](#). OnPutData event handler must send column data by [TDALoader.PutColumnData](#) method. TDALoader will flush data to Oracle when it is needed.

**See Also**

- [TDALoader.PutColumnData](#)
  - [Load](#)
  - [OnGetColumnData](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.7.2 Types

Types in the **DALoader** unit.

### Types

Name	Description
<a href="#">TDAPutDataEvent</a>	This type is used for the <a href="#">TDALoader.OnPutData</a> event.
<a href="#">TGetColumnDataEvent</a>	This type is used for the <a href="#">TDALoader.OnGetColumnData</a> event.
<a href="#">TLoaderProgressEvent</a>	This type is used for the <a href="#">TDALoader.OnProgress</a> event.

© 1997-2013 Devart. All Rights Reserved.

### 17.7.2.1 TDAPutDataEvent Procedure Reference

This type is used for the [TDALoader.OnPutData](#) event.

#### Unit

[DALoader](#)

#### Syntax

```
TDAPutDataEvent = procedure (Sender: TDALoader) of object;
```

#### Parameters

*Sender*

An object that raised the event.

© 1997-2013 Devart. All Rights Reserved.

### 17.7.2.2 TGetColumnDataEvent Procedure Reference

This type is used for the [TDALoader.OnGetColumnData](#) event.

#### Unit

[DALoader](#)

#### Syntax

```
TGetColumnDataEvent = procedure (Sender: TObject; Column: TDAColumn; Row: integer; var Value: variant; var IsEOF: boolean) of object;
```

#### Parameters

*Sender*

An object that raised the event.

*Column*

Points to [TDAColumn](#) object that corresponds to the current loading column.

*Row*

Indicates the current loading record.

*Value*

Holds column values.

*IsEOF*

True, if data loading needs to be stopped.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.7.2.3 TLoaderProgressEvent Procedure Reference

This type is used for the [TDALoader.OnProgress](#) event.

#### Unit

[DALoader](#)

#### Syntax

```
TLoaderProgressEvent = procedure (Sender: TObject; Percent: integer) of object;
```

#### Parameters

*Sender*

An object that raised the event.

*Percent*

Percentage of the load operation progress.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.8 DAScript

This unit contains the base class for the TIBCScript component.

#### Classes

Name	Description
<a href="#">TDAScript</a>	Makes it possible to execute several SQL statements one by one.
<a href="#">TDASStatement</a>	This class has attributes and methods for controlling single SQL statement of a script.
<a href="#">TDASStatements</a>	Holds a collection of <a href="#">TDASStatement</a> objects.

#### Types

Name	Description
<a href="#">TAfterStatementExecuteEvent</a>	This type is used for the <a href="#">TDAScript.AfterExecute</a> event.



[TBeforeStatementExecuteEvent](#)

This type is used for the [TDA Script.BeforeExecute](#) event.

[TOnErrorEvent](#)

This type is used for the [TDA Script.OnError](#) event.

## Enumerations

Name	Description
<a href="#">TErrorAction</a>	Indicates the action to take when the OnError handler exits.

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1 Classes

Classes in the **DAScript** unit.

### Classes

Name	Description
<a href="#">TDA Script</a>	Makes it possible to execute several SQL statements one by one.
<a href="#">TDA Statement</a>	This class has attributes and methods for controlling single SQL statement of a script.
<a href="#">TDA Statements</a>	Holds a collection of <a href="#">TDA Statement</a> objects.

© 1997-2013 Devart. All Rights Reserved.

### 17.8.1.1 TDA Script Class

Makes it possible to execute several SQL statements one by one.  
For a list of all members of this type, see [TDA Script](#) members.

### Unit

[DAScript](#)

### Syntax

```
TDA Script = class (TComponent);
```

### Remarks

Often it is necessary to execute several SQL statements one by one. This can be performed using a lot of components such as [TCustomDASQL](#) descendants. Usually it isn't the best solution. With only one TDA Script descendant component you can execute several SQL statements as one. This sequence of statements is called script. To separate single statements use semicolon (;) or slash (/) and for statements that can contain semicolon, (for example, CREATE TRIGGER or CREATE

PROCEDURE) only slash. Note that slash must be the first character in line. Errors that occur during execution can be processed in the [TDAScript.OnError](#) event handler. By default, on error TDAScript shows exception and continues execution.

## See Also

- [TCustomDASQL](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.8.1.1.1 Members

[TDAScript](#) class overview.

## Properties

Name	Description
<a href="#">Connection</a>	Used to specify the connection in which the script will be executed.
<a href="#">DataSet</a>	Refers to a dataset that holds the result set of query execution.
<a href="#">Debug</a>	Used to display the script execution and all its parameter values.
<a href="#">Delimiter</a>	Used to set the delimiter string that separates script statements.
<a href="#">EndLine</a>	Used to get the current statement last line number in a script.
<a href="#">EndOffset</a>	Used to get the offset in the last line of the current statement.
<a href="#">EndPos</a>	Used to get the end position of the current statement.
<a href="#">Macros</a>	Used to change SQL script text in design- or run-time easily.
<a href="#">SQL</a>	Used to get or set script text.
<a href="#">StartLine</a>	Used to get the current statement start line number in a script.
<a href="#">StartOffset</a>	Used to get the offset in the first line of the current statement.

[StartPos](#)

Used to get the start position of the current statement in a script.

[Statements](#)

Contains a list of statements obtained from the SQL property.

## Methods

Name	Description
<a href="#">BreakExec</a>	Stops script execution.
<a href="#">ErrorOffset</a>	Used to get the offset of the statement if the Execute method raised an exception.
<a href="#">Execute</a>	Executes a script.
<a href="#">ExecuteFile</a>	Executes SQL statements contained in a file.
<a href="#">ExecuteNext</a>	Executes the next statement in the script and then stops.
<a href="#">ExecuteStream</a>	Executes SQL statements contained in a stream object.
<a href="#">FindMacro</a>	Indicates whether a specified macro exists in a dataset.
<a href="#">MacroByName</a>	Finds a Macro with the name passed in Name.

## Events

Name	Description
<a href="#">AfterExecute</a>	Occurs after a SQL script execution.
<a href="#">BeforeExecute</a>	Occurs when taking a specific action before executing the current SQL statement is needed.
<a href="#">OnError</a>	Occurs when InterBase raises an error.

© 1997-2013 Devart. All Rights Reserved.

### 17.8.1.1.2 Properties

Properties of the **TDAScript** class.

For a complete list of the **TDAScript** class members, see the [TDAScript Members](#) topic.

## Public

Name	Description
------	-------------

[Connection](#)

Used to specify the connection in which the script will be executed.

[DataSet](#)

Refers to a dataset that holds the result set of query execution.

[EndLine](#)

Used to get the current statement last line number in a script.

[EndOffset](#)

Used to get the offset in the last line of the current statement.

[EndPos](#)

Used to get the end position of the current statement.

[StartLine](#)

Used to get the current statement start line number in a script.

[StartOffset](#)

Used to get the offset in the first line of the current statement.

[StartPos](#)

Used to get the start position of the current statement in a script.

[Statements](#)

Contains a list of statements obtained from the SQL property.

## Published

Name	Description
<a href="#">Debug</a>	Used to display the script execution and all its parameter values.
<a href="#">Delimiter</a>	Used to set the delimiter string that separates script statements.
<a href="#">Macros</a>	Used to change SQL script text in design- or run-time easily.
<a href="#">SQL</a>	Used to get or set script text.

## See Also

- [TDAScript Class](#)
- [TDAScript Class Members](#)

## 17.8.1.1.2.1 Connection Property

Used to specify the connection in which the script will be executed.

**Class**

[TDA Script](#)

**Syntax**

**property** Connection: [TCustomDAConnection](#);

**Remarks**

Use the Connection property to specify the connection in which the script will be executed. If Connection is not connected, the [Execute](#) method calls the Connect method of Connection.

Set at design-time by selecting from the list of provided [TCustomDAConnection](#) objects.

At run-time, set the Connection property to reference an existing TCustomDAConnection object.

**See Also**

- [TCustomDAConnection](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.1.2.2 DataSet Property

Refers to a dataset that holds the result set of query execution.

**Class**

[TDA Script](#)

**Syntax**

**property** DataSet: [TCustomDADataSet](#);

**Remarks**

Set the DataSet property to retrieve the results of the SELECT statements execution inside a script.

**See Also**

- [ExecuteNext](#)
- [Execute](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.1.2.3 Debug Property

Used to display the script execution and all its parameter values.

**Class**

[TDA Script](#)

**Syntax**

```
property Debug: boolean default False;
```

**Remarks**

Set the Debug property to True to display the script execution and all its parameter values. Also displays the type of parameters.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.1.2.4 Delimiter Property

Used to set the delimiter string that separates script statements.

**Class**

[TDA Script](#)

**Syntax**

```
property Delimiter: string stored IsDelimiterStored;
```

**Remarks**

Use the Delimiter property to set the delimiter string that separates script statements. By default it is semicolon (;). You can use slash (/) to separate statements that can contain semicolon (for example, CREATE TRIGGER or CREATE PROCEDURE) if the Delimiter property's default value is semicolon. Note that slash must be the first character in line.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.1.2.5 EndLine Property

Used to get the current statement last line number in a script.

**Class**

[TDA Script](#)

**Syntax**

```
property EndLine: Int64;
```

**Remarks**

Use the EndLine property to get the current statement last line number in a script.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.1.2.6 EndOffset Property

Used to get the offset in the last line of the current statement.

##### Class

[TDAScript](#)

##### Syntax

```
property EndOffset: Int64;
```

##### Remarks

Use the EndOffset property to get the offset in the last line of the current statement.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.1.2.7 EndPos Property

Used to get the end position of the current statement.

##### Class

[TDAScript](#)

##### Syntax

```
property EndPos: Int64;
```

##### Remarks

Use the EndPos property to get the end position of the current statement (the position of the last character in the statement) in a script.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.1.2.8 Macros Property

Used to change SQL script text in design- or run-time easily.

##### Class

[TDAScript](#)

##### Syntax

```
property Macros: TMacros stored False;
```

##### Remarks

With the help of macros you can easily change SQL script text in design- or run-time. Macros extend abilities of parameters and allow changing conditions in the WHERE clause or sort order in the ORDER BY clause. You just insert &MacroName in a SQL query text and change value of macro by the Macro property editor in design-time or the MacroByName function in run-time. In time of opening query macro is replaced by its value.

##### See Also

- [TMacro](#)
  - [MacroByName](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.1.2.9 SQL Property

Used to get or set script text.

#### **Class**

[TDAScript](#)

#### **Syntax**

**property** SQL: `_TStrings;`

#### **Remarks**

Use the SQL property to get or set script text.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.1.2.10 StartLine Property

Used to get the current statement start line number in a script.

#### **Class**

[TDAScript](#)

#### **Syntax**

**property** StartLine: `Int64;`

#### **Remarks**

Use the StartLine property to get the current statement start line number in a script.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.1.2.11 StartOffset Property

Used to get the offset in the first line of the current statement.

#### **Class**

[TDAScript](#)

#### **Syntax**

**property** StartOffset: `Int64;`

#### **Remarks**

Use the StartOffset property to get the offset in the first line of the current statement.

---

© 1997-2013 Devart. All Rights Reserved.



## 17.8.1.1.2.12 StartPos Property

Used to get the start position of the current statement in a script.

**Class**

[TDAScript](#)

**Syntax**

```
property StartPos: Int64;
```

**Remarks**

Use the StartPos property to get the start position of the current statement (the position of the first statement character) in a script.

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.1.2.13 Statements Property

Contains a list of statements obtained from the SQL property.

**Class**

[TDAScript](#)

**Syntax**

```
property Statements: TDASentences;
```

**Remarks**

Contains a list of statements that are obtained from the SQL property. Use the Access Statements property to view SQL statement, set parameters or execute the specified statement. Statements is a zero-based array of statement records. Index specifies the array element to access.

For example, consider the following script:

```
CREATE TABLE A (FIELD1 INTEGER);  
INSERT INTO A VALUES (1);  
INSERT INTO A VALUES (2);  
INSERT INTO A VALUES (3);  
CREATE TABLE B (FIELD1 INTEGER);  
INSERT INTO B VALUES (1);  
INSERT INTO B VALUES (2);  
INSERT INTO B VALUES (3);
```

**Note:** The list of statements is created and filled when the value of Statements property is requested. That's why the first access to the Statements property can take a long time.

**Example**

You can use the Statements property in the following way:

```
procedure TForm1.Button1Click(Sender: TObject);  
var  
    i: integer;  
begin  
    with Script do  
        begin
```

```

    for i := 0 to Statements.Count - 1 do
        if Copy(Statements[i].SQL, 1, 6) <> 'CREATE' then
            Statements[i].Execute;
        end;
    end;

```

## See Also

- [TDASentences](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.8.1.1.3 Methods

Methods of the **TDAScript** class.

For a complete list of the **TDAScript** class members, see the [TDAScript Members](#) topic.

## Public

Name	Description
<a href="#">BreakExec</a>	Stops script execution.
<a href="#">ErrorOffset</a>	Used to get the offset of the statement if the Execute method raised an exception.
<a href="#">Execute</a>	Executes a script.
<a href="#">ExecuteFile</a>	Executes SQL statements contained in a file.
<a href="#">ExecuteNext</a>	Executes the next statement in the script and then stops.
<a href="#">ExecuteStream</a>	Executes SQL statements contained in a stream object.
<a href="#">FindMacro</a>	Indicates whether a specified macro exists in a dataset.
<a href="#">MacroByName</a>	Finds a Macro with the name passed in Name.

## See Also

- [TDAScript Class](#)
- [TDAScript Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.8.1.1.3.1 BreakExec Method

Stops script execution.

## Class

[TDAScript](#)

**Syntax**

```
procedure BreakExec; virtual;
```

**Remarks**

Call the BreakExec method to stop script execution.

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.1.3.2 ErrorOffset Method

Used to get the offset of the statement if the Execute method raised an exception.

**Class**

[TDA Script](#)

**Syntax**

```
function ErrorOffset: Int64;
```

**Return Value**

offset of an error.

**Remarks**

Call the ErrorOffset method to get the offset of the statement if the Execute method raised an exception.

**See Also**

- [OnError](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.1.3.3 Execute Method

Executes a script.

**Class**

[TDA Script](#)

**Syntax**

```
procedure Execute; virtual;
```

**Remarks**

Call the Execute method to execute a script. If InterBase raises an error, the OnError event occurs.

**See Also**

- [ExecuteNext](#)

- [OnError](#)
  - [ErrorOffset](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.1.3.4 ExecuteFile Method

Executes SQL statements contained in a file.

#### Class

[TDAScript](#)

#### Syntax

```
procedure ExecuteFile(const FileName: string) ;
```

#### Parameters

*FileName*

Holds the file name.

#### Remarks

Call the ExecuteFile method to execute SQL statements contained in a file. Script doesn't load full content into memory. Reading and execution is performed by blocks of 64k size. Therefore, it is optimal to use it for big files.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.1.3.5 ExecuteNext Method

Executes the next statement in the script and then stops.

#### Class

[TDAScript](#)

#### Syntax

```
function ExecuteNext: boolean; virtual;
```

#### Return Value

True, if there are any statements left in the script, False otherwise.

#### Remarks

Use the ExecuteNext method to execute the next statement in the script statement and stop. If InterBase raises an error, the OnError event occurs.

#### See Also

- [Execute](#)
  - [OnError](#)
  - [ErrorOffset](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.1.3.6 ExecuteStream Method

Executes SQL statements contained in a stream object.

**Class**

[TDA Script](#)

**Syntax**

```
procedure ExecuteStream(Stream: TStream);
```

**Parameters**

*Stream*

Holds the stream object from which the statements will be executed.

**Remarks**

Call the ExecuteStream method to execute SQL statements contained in a stream object. Reading from the stream and execution is performed by blocks of 64k size.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.1.3.7 FindMacro Method

Indicates whether a specified macro exists in a dataset.

**Class**

[TDA Script](#)

**Syntax**

```
function FindMacro(Name: string): TMacro;
```

**Parameters**

*Name*

Holds the name of the macro to search for.

**Return Value**

a TMacro object, if a macro with matching name was found, otherwise returns nil.

**Remarks**

Call the FindMacro method to determine if a specified macro exists. If FindMacro finds a macro with a matching name, it returns a TMacro object for the specified Name. Otherwise it returns nil.

**See Also**

- [TMacro](#)
- [Macros](#)
- [MacroByName](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.1.3.8 MacroByName Method

Finds a Macro with the name passed in Name.

**Class**

[TDA Script](#)

**Syntax**

```
function MacroByName (Name: string) : TMacro;
```

**Parameters**

*Name*

Holds the name of the Macro to search for.

**Return Value**

the Macro, if a match was found.

**Remarks**

Call the MacroByName method to find a Macro with the name passed in Name. If a match was found, MacroByName returns the Macro. Otherwise, an exception is raised. Use this method rather than a direct reference to the Items property to avoid depending on the order of the entries.

To locate a parameter by name without raising an exception if the parameter is not found, use the FindMacro method.

To assign the value of macro use the [TMacro.Value](#) property.

**See Also**

- [TMacro](#)
  - [Macros](#)
  - [FindMacro](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.1.4 Events

Events of the **TDA Script** class.

For a complete list of the **TDA Script** class members, see the [TDA Script Members](#) topic.

**Published**

Name	Description
<a href="#">AfterExecute</a>	Occurs after a SQL script execution.
<a href="#">BeforeExecute</a>	Occurs when taking a specific action before executing the current SQL statement is needed.
<a href="#">OnError</a>	Occurs when InterBase raises an error.

**See Also**

- [TDA Script Class](#)
- [TDA Script Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

**17.8.1.1.4.1 AfterExecute Event**

Occurs after a SQL script execution.

**Class**

[TDA Script](#)

**Syntax**

**property** AfterExecute: [TAfterStatementExecuteEvent](#);

**Remarks**

Occurs after a SQL script has been executed.

**See Also**

- [Execute](#)

---

© 1997-2013 Devart. All Rights Reserved.

**17.8.1.1.4.2 BeforeExecute Event**

Occurs when taking a specific action before executing the current SQL statement is needed.

**Class**

[TDA Script](#)

**Syntax**

**property** BeforeExecute: [TBeforeStatementExecuteEvent](#);

**Remarks**

Write the BeforeExecute event handler to take specific action before executing the current SQL statement. SQL holds text of the current SQL statement. Write SQL to change the statement that will be executed. Set Omit to True to skip statement execution.

---

© 1997-2013 Devart. All Rights Reserved.

**17.8.1.1.4.3 OnError Event**

Occurs when InterBase raises an error.

**Class**

[TDA Script](#)

**Syntax**

```
property OnError: TOnErrorEvent;
```

**Remarks**

Occurs when InterBase raises an error.

Action indicates the action to take when the OnError handler exits. On entry into the handler, Action is always set to eaFail.

**See Also**

- [ErrorOffset](#)
- 

© 1997-2013 Devart. All Rights Reserved.

**17.8.1.2 TDASentence Class**

This class has attributes and methods for controlling single SQL statement of a script.

For a list of all members of this type, see [TDASentence](#) members.

**Unit**

[DAScript](#)

**Syntax**

```
TDASentence = class(TCollectionItem);
```

**Remarks**

DAScript contains SQL statements, represented as TDASentence objects. The TDASentence class has attributes and methods for controlling single SQL statement of a script.

**See Also**

- [DAScript](#)
  - [TDASentences](#)
- 

© 1997-2013 Devart. All Rights Reserved.

**17.8.1.2.1 Members**

[TDASentence](#) class overview.

**Properties**

Name	Description
<a href="#">EndLine</a>	Used to determine the number of the last statement line in a script.



<a href="#">EndOffset</a>	Used to get the offset in the last line of the statement.
<a href="#">EndPos</a>	Used to get the end position of the statement in a script.
<a href="#">Omit</a>	Used to avoid execution of a statement.
<a href="#">Params</a>	Contains parameters for an SQL statement.
<a href="#">Script</a>	Used to determine the TDA Script object the SQL Statement belongs to.
<a href="#">SQL</a>	Used to get or set the text of an SQL statement.
<a href="#">StartLine</a>	Used to determine the number of the first statement line in a script.
<a href="#">StartOffset</a>	Used to get the offset in the first line of a statement.
<a href="#">StartPos</a>	Used to get the start position of the statement in a script.

## Methods

Name	Description
<a href="#">Execute</a>	Executes a statement.

© 1997-2013 Devart. All Rights Reserved.

### 17.8.1.2.2 Properties

Properties of the **TDAStatement** class.

For a complete list of the **TDAStatement** class members, see the [TDAStatement Members](#) topic.

## Public

Name	Description
<a href="#">EndLine</a>	Used to determine the number of the last statement line in a script.
<a href="#">EndOffset</a>	Used to get the offset in the last line of the statement.
<a href="#">EndPos</a>	Used to get the end position of the statement in a script.
<a href="#">Omit</a>	Used to avoid execution of a statement.
<a href="#">Params</a>	Contains parameters for an SQL statement.

[Script](#)

Used to determine the TDA Script object the SQL Statement belongs to.

[SQL](#)

Used to get or set the text of an SQL statement.

[StartLine](#)

Used to determine the number of the first statement line in a script.

[StartOffset](#)

Used to get the offset in the first line of a statement.

[StartPos](#)

Used to get the start position of the statement in a script.

### See Also

- [TDAStatement Class](#)
  - [TDAStatement Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.2.2.1 EndLine Property

Used to determine the number of the last statement line in a script.

### Class

[TDAStatement](#)

### Syntax

```
property EndLine: integer;
```

### Remarks

Use the EndLine property to determine the number of the last statement line in a script.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.2.2.2 EndOffset Property

Used to get the offset in the last line of the statement.

### Class

[TDAStatement](#)

### Syntax

```
property EndOffset: integer;
```

### Remarks

Use the EndOffset property to get the offset in the last line of the statement.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.2.2.3 EndPos Property

Used to get the end position of the statement in a script.

**Class**

[TDASentence](#)

**Syntax**

```
property EndPos: integer;
```

**Remarks**

Use the EndPos property to get the end position of the statement (the position of the last character in the statement) in a script.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.2.2.4 Omit Property

Used to avoid execution of a statement.

**Class**

[TDASentence](#)

**Syntax**

```
property Omit: boolean;
```

**Remarks**

Set the Omit property to True to avoid execution of a statement.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.2.2.5 Params Property

Contains parameters for an SQL statement.

**Class**

[TDASentence](#)

**Syntax**

```
property Params: TDAParams;
```

**Remarks**

Contains parameters for an SQL statement.

Access Params at runtime to view and set parameter names, values, and data types dynamically. Params is a zero-based array of parameter records. Index specifies the array element to access.

**See Also**

- [TDAParam](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.2.2.6 Script Property

Used to determine the TDA Script object the SQL Statement belongs to.

### Class

[TDAScript](#)

### Syntax

```
property Script: TDAScript;
```

### Remarks

Use the Script property to determine the TDA Script object the SQL Statement belongs to.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.2.2.7 SQL Property

Used to get or set the text of an SQL statement.

### Class

[TDAScript](#)

### Syntax

```
property SQL: string;
```

### Remarks

Use the SQL property to get or set the text of an SQL statement.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.1.2.2.8 StartLine Property

Used to determine the number of the first statement line in a script.

### Class

[TDAScript](#)

### Syntax

```
property StartLine: integer;
```

### Remarks

Use the StartLine property to determine the number of the first statement line in a script.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.2.2.9 StartOffset Property

Used to get the offset in the first line of a statement.

**Class**

[TDASatement](#)

**Syntax**

```
property StartOffset: integer;
```

**Remarks**

Use the StartOffset property to get the offset in the first line of a statement.

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.2.2.10 StartPos Property

Used to get the start position of the statement in a script.

**Class**

[TDASatement](#)

**Syntax**

```
property StartPos: integer;
```

**Remarks**

Use the StartPos property to get the start position of the statement (the position of the first statement character) in a script.

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.2.3 Methods

Methods of the **TDASatement** class.

For a complete list of the **TDASatement** class members, see the [TDASatement Members](#) topic.

**Public**

Name	Description
<a href="#">Execute</a>	Executes a statement.

**See Also**

- [TDASatement Class](#)
- [TDASatement Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.8.1.2.3.1 Execute Method

Executes a statement.

**Class**

[TDASentence](#)**Syntax**

```
procedure Execute;
```

**Remarks**

Use the Execute method to execute a statement.

---

© 1997-2013 Devart. All Rights Reserved.

**17.8.1.3 TDASentences Class**

Holds a collection of [TDASentence](#) objects.

For a list of all members of this type, see [TDASentences](#) members.

**Unit**

[DAScript](#)

**Syntax**

```
TDASentences = class(TCollection);
```

**Remarks**

Each TDASentences holds a collection of [TDASentence](#) objects. TDASentences maintains an index of the sentences in its Items array. The Count property contains the number of sentences in the collection. Use TDASentences class to manipulate script SQL sentences.

**See Also**

- [DAScript](#)
  - [TDASentence](#)
- 

© 1997-2013 Devart. All Rights Reserved.

**17.8.1.3.1 Members**

[TDASentences](#) class overview.

**Properties****Name**

[Items](#)

**Description**

Used to access separate script sentences.

---

© 1997-2013 Devart. All Rights Reserved.

**17.8.1.3.2 Properties**

Properties of the **TDASentences** class.

For a complete list of the **TDASentences** class members, see the [TDASentences Members](#) topic.

**Public****Name**[Items](#)**Description**

Used to access separate script statements.

**See Also**

- [TDAStatements Class](#)
- [TDAStatements Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

**17.8.1.3.2.1 Items Property(Indexer)**

Used to access separate script statements.

**Class**[TDAStatements](#)**Syntax**

```
property Items[Index: Integer]: TDASatement; default;
```

**Parameters***Index*

Holds the index value.

**Remarks**

Use the Items property to access individual script statements. The value of the Index parameter corresponds to the Index property of [TDASatement](#).

**See Also**

- [TDASatement](#)

© 1997-2013 Devart. All Rights Reserved.

**17.8.2 Types**

Types in the **DAScript** unit.

**Types****Name**[TAfterStatementExecuteEvent](#)**Description**

This type is used for the [TDAScript.AfterExecute](#) event.

[TBeforeStatementExecuteEvent](#)

This type is used for the [TDAScript.BeforeExecute](#) event.

[TOnErrorEvent](#)

This type is used for the [TDAScript.OnError](#) event.

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.2.1 TAfterStatementExecuteEvent Procedure Reference

This type is used for the [TDA Script.AfterExecute](#) event.

##### Unit

[DAScript](#)

##### Syntax

```
TAfterStatementExecuteEvent = procedure (Sender: TObject; SQL: string) of object;
```

##### Parameters

*Sender*

An object that raised the event.

*SQL*

Holds the passed SQL statement.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.2.2 TBeforeStatementExecuteEvent Procedure Reference

This type is used for the [TDA Script.BeforeExecute](#) event.

##### Unit

[DAScript](#)

##### Syntax

```
TBeforeStatementExecuteEvent = procedure (Sender: TObject; var SQL: string; var Omit: boolean) of object;
```

##### Parameters

*Sender*

An object that raised the event.

*SQL*

Holds the passed SQL statement.

*Omit*

True, if the statement execution should be skipped.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.2.3 TOnErrorEvent Procedure Reference

This type is used for the [TDA Script.OnError](#) event.

##### Unit

[DAScript](#)

##### Syntax

```
TOnErrorEvent = procedure (Sender: TObject; E: Exception; SQL:
```

---



```
string; var Action: TErrorAction) of object;
```

**Parameters***Sender*

An object that raised the event.

*E*

The error code.

*SQL*

Holds the passed SQL statement.

*Action*

The action to take when the OnError handler exits.

© 1997-2013 Devart. All Rights Reserved.

### 17.8.3 Enumerations

Enumerations in the **DAScript** unit.

**Enumerations**

Name	Description
<a href="#">TErrorAction</a>	Indicates the action to take when the OnError handler exits.

© 1997-2013 Devart. All Rights Reserved.

#### 17.8.3.1 TErrorAction Enumeration

Indicates the action to take when the OnError handler exits.

**Unit**

[DAScript](#)

**Syntax**

```
TErrorAction = (eaAbort, eaFail, eaException, eaContinue);
```

**Values**

Value	Meaning
<b>eaAbort</b>	Abort execution without displaying an error message.
<b>eaContinue</b>	Continue execution.
<b>eaException</b>	In Delphi 6 and higher exception is handled by the Application.HandleException method.
<b>eaFail</b>	Abort execution and display an error message.

© 1997-2013 Devart. All Rights Reserved.

## 17.9 DASQLMonitor

This unit contains the base class for the TIBCSQLMonitor component.

### Classes

Name	Description
<a href="#">TCustomDASQLMonitor</a>	A base class that introduces properties and methods to monitor dynamic SQL execution in database applications interactively.
<a href="#">TDBMonitorOptions</a>	This class holds options for dbMonitor.

### Types

Name	Description
<a href="#">TDATraceFlags</a>	Represents the set of <a href="#">TDATraceFlag</a> .
<a href="#">TMonitorOptions</a>	Represents the set of <a href="#">TMonitorOption</a> .
<a href="#">TOnSQLEvent</a>	This type is used for the <a href="#">TCustomDASQLMonitor.OnSQL</a> event.

### Enumerations

Name	Description
<a href="#">TDATraceFlag</a>	Use TraceFlags to specify which database operations the monitor should track in an application at runtime.
<a href="#">TMonitorOption</a>	Used to define where information from SQLMonitor will be dispalyed.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.9.1 Classes

Classes in the **DASQLMonitor** unit.

### Classes

Name	Description
<a href="#">TCustomDASQLMonitor</a>	A base class that introduces properties and methods to monitor dynamic SQL execution in database applications interactively.
<a href="#">TDBMonitorOptions</a>	This class holds options for dbMonitor.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.9.1.1 TCustomDASQLMonitor Class

A base class that introduces properties and methods to monitor dynamic SQL execution in database applications interactively.

For a list of all members of this type, see [TCustomDASQLMonitor](#) members.

#### Unit

[DASQLMonitor](#)

#### Syntax

```
TCustomDASQLMonitor = class (TComponent);
```

#### Remarks

TCustomDASQLMonitor is a base class that introduces properties and methods to monitor dynamic SQL execution in database applications interactively.

TCustomDASQLMonitor provides two ways of displaying debug information. It monitors either by dialog window or by Borland's proprietary SQL Monitor.

Furthermore to receive debug information use the [TCustomDASQLMonitor.OnSQL](#) event.

In applications use descendants of TCustomDASQLMonitor.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.1.1.1 Members

[TCustomDASQLMonitor](#) class overview.

#### Properties

Name	Description
<a href="#">Active</a>	Used to activate monitoring of SQL.
<a href="#">DBMonitorOptions</a>	Used to set options for dbMonitor.
<a href="#">Options</a>	Used to include the desired properties for TCustomDASQLMonitor.
<a href="#">TraceFlags</a>	Used to specify which database operations the monitor should track in an application at runtime.

#### Events

Name	Description
<a href="#">OnSQL</a>	Occurs when tracing of SQL activity on database components is needed.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.9.1.1.2 Properties

Properties of the **TCustomDASQLMonitor** class.

For a complete list of the **TCustomDASQLMonitor** class members, see the [TCustomDASQLMonitor Members](#) topic.

**Public**

Name	Description
<a href="#">Active</a>	Used to activate monitoring of SQL.
<a href="#">DBMonitorOptions</a>	Used to set options for dbMonitor.
<a href="#">Options</a>	Used to include the desired properties for TCustomDASQLMonitor.
<a href="#">TraceFlags</a>	Used to specify which database operations the monitor should track in an application at runtime.

**See Also**

- [TCustomDASQLMonitor Class](#)
  - [TCustomDASQLMonitor Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.9.1.1.2.1 Active Property

Used to activate monitoring of SQL.

**Class**

[TCustomDASQLMonitor](#)

**Syntax**

```
property Active: boolean default True;
```

**Remarks**

Set the Active property to True to activate monitoring of SQL.

**See Also**

- [OnSQL](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.9.1.1.2.2 DBMonitorOptions Property

Used to set options for dbMonitor.

**Class**

[TCustomDASQLMonitor](#)

### Syntax

```
property DBMonitorOptions: TDBMonitorOptions;
```

### Remarks

Use DBMonitorOptions to set options for dbMonitor.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.1.1.2.3 Options Property

Used to include the desired properties for TCustomDASQLMonitor.

### Class

[TCustomDASQLMonitor](#)

### Syntax

```
property Options: TMonitorOptions default [moDialog, moSQLMonitor,  
moDBMonitor, moCustom];
```

### Remarks

Set Options to include the desired properties for TCustomDASQLMonitor.

### See Also

- [OnSQL](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.1.1.2.4 TraceFlags Property

Used to specify which database operations the monitor should track in an application at runtime.

### Class

[TCustomDASQLMonitor](#)

### Syntax

```
property TraceFlags: TDATraceFlags default [tfQPrepare,  
tfQExecute, tfError, tfConnect, tfTransact, tfParams, tfMisc];
```

### Remarks

Use the TraceFlags property to specify which database operations the monitor should track in an application at runtime.

### See Also

- [OnSQL](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.1.1.3 Events

Events of the **TCustomDASQLMonitor** class.

For a complete list of the **TCustomDASQLMonitor** class members, see the [TCustomDASQLMonitor Members](#) topic.

#### Public

Name	Description
<a href="#">OnSQL</a>	Occurs when tracing of SQL activity on database components is needed.

#### See Also

- [TCustomDASQLMonitor Class](#)
  - [TCustomDASQLMonitor Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.1.1.3.1 OnSQL Event

Occurs when tracing of SQL activity on database components is needed.

#### Class

[TCustomDASQLMonitor](#)

#### Syntax

**property** OnSQL: [TOnSQLEvent](#);

#### Remarks

Write the OnSQL event handler to let an application trace SQL activity on database components. The Text parameter holds the detected SQL statement. Use the Flag parameter to make selective processing of SQL in the handler body.

#### See Also

- [TraceFlags](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.1.2 TDBMonitorOptions Class

This class holds options for dbMonitor.

For a list of all members of this type, see [TDBMonitorOptions](#) members.

#### Unit

[DASQLMonitor](#)

#### Syntax

```
TDBMonitorOptions = class (TPersistent);
```

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.1.2.1 Members

[TDBMonitorOptions](#) class overview.

### Properties

Name	Description
<a href="#">Host</a>	Used to set the host name or IP address of the computer where dbMonitor application runs.
<a href="#">Port</a>	Used to set the port number for connecting to dbMonitor.
<a href="#">ReconnectTimeout</a>	Used to set the minimum time that should be spent before reconnecting to dbMonitor is allowed.
<a href="#">SendTimeout</a>	Used to set timeout for sending events to dbMonitor.

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.1.2.2 Properties

Properties of the **TDBMonitorOptions** class.

For a complete list of the **TDBMonitorOptions** class members, see the [TDBMonitorOptions Members](#) topic.

### Published

Name	Description
<a href="#">Host</a>	Used to set the host name or IP address of the computer where dbMonitor application runs.
<a href="#">Port</a>	Used to set the port number for connecting to dbMonitor.
<a href="#">ReconnectTimeout</a>	Used to set the minimum time that should be spent before reconnecting to dbMonitor is allowed.
<a href="#">SendTimeout</a>	Used to set timeout for sending events to dbMonitor.

### See Also

- [TDBMonitorOptions Class](#)
- [TDBMonitorOptions Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.1.2.2.1 Host Property

Used to set the host name or IP address of the computer where dbMonitor application runs.

##### **Class**

[TDBMonitorOptions](#)

##### **Syntax**

```
property Host: string;
```

##### **Remarks**

Use the Host property to set the host name or IP address of the computer where dbMonitor application runs.  
dbMonitor supports remote monitoring. You can run dbMonitor on a different computer than monitored application runs. In this case you need to set the Host property to the corresponding computer name.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.1.2.2.2 Port Property

Used to set the port number for connecting to dbMonitor.

##### **Class**

[TDBMonitorOptions](#)

##### **Syntax**

```
property Port: integer default DBMonitorPort;
```

##### **Remarks**

Use the Port property to set the port number for connecting to dbMonitor.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.1.2.2.3 ReconnectTimeout Property

Used to set the minimum time that should be spent before reconnecting to dbMonitor is allowed.

##### **Class**

[TDBMonitorOptions](#)

##### **Syntax**

```
property ReconnectTimeout: integer default  
DefaultReconnectTimeout;
```

##### **Remarks**

Use the ReconnectTimeout property to set the minimum time (in milliseconds) that



should be spent before allowing reconnecting to dbMonitor. If an error occurs when the component sends an event to dbMonitor (dbMonitor is not running), next events are ignored and the component does not restore the connection until ReconnectTimeout is over.

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.1.2.2.4 SendTimeout Property

Used to set timeout for sending events to dbMonitor.

### Class

[TDBMonitorOptions](#)

### Syntax

```
property SendTimeout: integer default DefaultSendTimeout;
```

### Remarks

Use the SendTimeout property to set timeout (in milliseconds) for sending events to dbMonitor. If dbMonitor does not respond in the specified timeout, event is ignored.

© 1997-2013 Devart. All Rights Reserved.

## 17.9.2 Types

Types in the **DASQLMonitor** unit.

### Types

Name	Description
<a href="#">TDATraceFlags</a>	Represents the set of <a href="#">TDATraceFlag</a> .
<a href="#">TMonitorOptions</a>	Represents the set of <a href="#">TMonitorOption</a> .
<a href="#">TOnSQLEvent</a>	This type is used for the <a href="#">TCustomDASQLMonitor.OnSQL</a> event.

© 1997-2013 Devart. All Rights Reserved.

#### 17.9.2.1 TDATraceFlags Set

Represents the set of [TDATraceFlag](#).

### Unit

[DASQLMonitor](#)

### Syntax

```
TDATraceFlags = set of TDATraceFlag;
```

© 1997-2013 Devart. All Rights Reserved.

### 17.9.2.2 TMonitorOptions Set

Represents the set of [TMonitorOption](#).

#### Unit

[DASQLMonitor](#)

#### Syntax

```
TMonitorOptions = set of TMonitorOption;
```

---

© 1997-2013 Devart. All Rights Reserved.

### 17.9.2.3 TOnSQLEvent Procedure Reference

This type is used for the [TCustomDASQLMonitor.OnSQL](#) event.

#### Unit

[DASQLMonitor](#)

#### Syntax

```
TOnSQLEvent = procedure (Sender: TObject; Text: string; Flag: TDATraceFlag) of object;
```

#### Parameters

##### *Sender*

An object that raised the event.

##### *Text*

Holds the detected SQL statement.

##### *Flag*

Use the Flag parameter to make selective processing of SQL in the handler body.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.9.3 Enumerations

Enumerations in the **DASQLMonitor** unit.

#### Enumerations

Name	Description
<a href="#">TDATraceFlag</a>	Use TraceFlags to specify which database operations the monitor should track in an application at runtime.
<a href="#">TMonitorOption</a>	Used to define where information from SQLMonitor will be dispalyed.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.9.3.1 TDATraceFlag Enumeration

Use TraceFlags to specify which database operations the monitor should track in an application at runtime.

#### Unit

[DASQLMonitor](#)

#### Syntax

```
TDATraceFlag = (tfQPrepare, tfQExecute, tfQFetch, tfError, tfStmt,  
tfConnect, tfTransact, tfBlob, tfService, tfMisc, tfParams,  
tfObjDestroy, tfPool);
```

#### Values

Value	Meaning
<b>tfBlob</b>	This option is declared for future use.
<b>tfConnect</b>	Establishing a connection.
<b>tfError</b>	Errors of query execution.
<b>tfMisc</b>	This option is declared for future use.
<b>tfObjDestroy</b>	Destroying of components.
<b>tfParams</b>	Representing parameter values for tfQPrepare and tfQExecute.
<b>tfPool</b>	Connection pool operations.
<b>tfQExecute</b>	Execution of the queries.
<b>tfQFetch</b>	This option is declared for future use.
<b>tfQPrepare</b>	Queries preparation.
<b>tfService</b>	This option is declared for future use.
<b>tfStmt</b>	This option is declared for future use.
<b>tfTransact</b>	Processing transactions.

© 1997-2013 Devart. All Rights Reserved.

### 17.9.3.2 TMonitorOption Enumeration

Used to define where information from SQLMonitor will be displayed.

#### Unit

[DASQLMonitor](#)

#### Syntax

```
TMonitorOption = (moDialog, moSQLMonitor, moDBMonitor, moCustom,  
moHandled);
```

#### Values

Value	Meaning
<b>moCustom</b>	Monitoring of SQL for individual components is allowed. Set Debug properties in SQL-related components to True to let TCustomDASQLMonitor instance to monitor their behavior. Has effect when moDialog is included.

<b>moDBMonitor</b>	Debug information is displayed in A:Using DBMonitor.
<b>moDialog</b>	Debug information is displayed in debug window.
<b>moHandled</b>	Component handle is included into the event description string.
<b>moSQLMonitor</b>	Debug information is displayed in Borland SQL Monitor.

© 1997-2013 Devart. All Rights Reserved.

## 17.10 DBAccess

This unit contains base classes for most of the components.

### Classes

Name	Description
<a href="#">EDAEError</a>	A base class for exceptions that are raised when an error occurs on the server side.
<a href="#">TCRDataSource</a>	Provides an interface between a DAC dataset components and data-aware controls on a form.
<a href="#">TCustomConnectDialog</a>	A base class for the connect dialog components.
<a href="#">TCustomDACConnection</a>	A base class for components used to establish connections.
<a href="#">TCustomDADataSet</a>	Encapsulates general set of properties, events, and methods for working with data accessed through various database engines.
<a href="#">TCustomDASQL</a>	A base class for components executing SQL statements that do not return result sets.
<a href="#">TCustomDAUpdateSQL</a>	A base class for components that provide DML statements for more flexible control over data modifications.
<a href="#">TDACConnectionOptions</a>	This class allows setting up the behaviour of the TDACConnection class.
<a href="#">TDADatasetOptions</a>	This class allows setting up the behaviour of the TDADataset class.
<a href="#">TDAEncryptionOptions</a>	Used to specify the options of the data encryption in a dataset.

[TDAMapRule](#)

Class that formes rules for Data Type Mapping.

[TDAMapRules](#)

Used for adding rules for DataSet fields mapping with both identifying by field name and by field type and Delphi field types.

[TDAMetaData](#)

A class for retrieving metainformation of the specified database objects in the form of dataset.

[TDAParam](#)

A class that forms objects to represent the values of the [parameters set](#).

[TDAParams](#)

This class is used to manage a list of TDAParam objects for an object that uses field parameters.

[TDATransaction](#)

A base class that implements functionality for controlling transactions.

[TMacro](#)

Object that represents the value of a macro.

[TMacros](#)

Controls a list of TMacro objects for the [TCustomDASQL.Macros](#) or [TCustomDADataset](#) components.

[TPoolingOptions](#)

This class allows setting up the behaviour of the connection pool.

## Types

Name	Description
<a href="#">TAfterExecuteEvent</a>	This type is used for the <a href="#">TCustomDADataset.AfterExecute</a> and <a href="#">TCustomDASQL.AfterExecute</a> events.
<a href="#">TAfterFetchEvent</a>	This type is used for the <a href="#">TCustomDADataset.AfterFetch</a> event.
<a href="#">TBeforeFetchEvent</a>	This type is used for the <a href="#">TCustomDADataset.BeforeFetch</a> event.
<a href="#">TConnectionLostEvent</a>	This type is used for the <a href="#">TCustomDAConnection.OnConnectionLost</a> event.

[TDAConnectionErrorEvent](#)

This type is used for the [TCustomDAConnection.OnError](#) event.

[TDATransactionErrorEvent](#)

This type is used for the [TDATransaction.OnError](#) event.

[TRefreshOptions](#)

Represents the set of [TRefreshOption](#).

[TUpdateExecuteEvent](#)

This type is used for the TCustomDADataset.  
AfterUpdateExecute and  
TCustomDADataset.  
BeforeUpdateExecute  
events.

## Enumerations

Name	Description
<a href="#">TLabelSet</a>	Sets the language of labels in the connect dialog.
<a href="#">TLockMode</a>	Specifies when to perform locking of an editing record.
<a href="#">TRefreshOption</a>	Indicates when the editing record will be refreshed.
<a href="#">TRetryMode</a>	Specifies the application behavior when connection is lost.

## Variables

Name	Description
<a href="#">ChangeCursor</a>	When set to True allows data access components to change screen cursor for the execution time.
<a href="#">MacroChar</a>	Determinates what character is used for macros.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1 Classes

Classes in the **DBAccess** unit.

## Classes

Name	Description
<a href="#">EDAEError</a>	A base class for exceptions that are raised when an error occurs on the server side.

<a href="#"><u>TCRDataSource</u></a>	Provides an interface between a DAC dataset components and data-aware controls on a form.
<a href="#"><u>TCustomConnectDialog</u></a>	A base class for the connect dialog components.
<a href="#"><u>TCustomDAConnection</u></a>	A base class for components used to establish connections.
<a href="#"><u>TCustomDADataSet</u></a>	Encapsulates general set of properties, events, and methods for working with data accessed through various database engines.
<a href="#"><u>TCustomDASQL</u></a>	A base class for components executing SQL statements that do not return result sets.
<a href="#"><u>TCustomDAUpdateSQL</u></a>	A base class for components that provide DML statements for more flexible control over data modifications.
<a href="#"><u>TDAConnectionOptions</u></a>	This class allows setting up the behaviour of the TDAConnection class.
<a href="#"><u>TDADatasetOptions</u></a>	This class allows setting up the behaviour of the TDADataset class.
<a href="#"><u>TDASecurityOptions</u></a>	Used to specify the options of the data encryption in a dataset.
<a href="#"><u>TDAMapRule</u></a>	Class that forms rules for Data Type Mapping.
<a href="#"><u>TDAMapRules</u></a>	Used for adding rules for DataSet fields mapping with both identifying by field name and by field type and Delphi field types.
<a href="#"><u>TDAMetaData</u></a>	A class for retrieving meta-information of the specified database objects in the form of dataset.
<a href="#"><u>TDAParam</u></a>	A class that forms objects to represent the values of the <a href="#"><u>parameters set</u></a> .
<a href="#"><u>TDAParams</u></a>	This class is used to manage a list of TDAParam objects for an object that uses field parameters.

[TDATransaction](#)

A base class that implements functionality for controlling transactions.

[TMacro](#)

Object that represents the value of a macro.

[TMacros](#)

Controls a list of TMacro objects for the [TCustomDASQL.Macros](#) or [TCustomDADataset](#) components.

[TPoolingOptions](#)

This class allows setting up the behaviour of the connection pool.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.1 EDAError Class

A base class for exceptions that are raised when an error occurs on the server side. For a list of all members of this type, see [EDAError](#) members.

#### Unit

[DBAccess](#)

#### Syntax

```
EDAError = class (EDatabaseError);
```

#### Remarks

EDAError is a base class for exceptions that are raised when an error occurs on the server side.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.1.1 Members

[EDAError](#) class overview.

#### Properties

Name	Description
<a href="#">Component</a>	Contains the component that caused the error.
<a href="#">ErrorCode</a>	Determines the error code returned by the server.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.1.2 Properties

Properties of the **EDAError** class.  
For a complete list of the **EDAError** class members, see the [EDAError Members](#) topic.



**Public****Name**[Component](#)[ErrorCode](#)**Description**

Contains the component that caused the error.

Determines the error code returned by the server.

**See Also**

- [EDAEError Class](#)
- [EDAEError Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

**17.10.1.1.2.1 Component Property**

Contains the component that caused the error.

**Class**[EDAEError](#)**Syntax**

```
property Component: TObject;
```

**Remarks**

The Component property contains the component that caused the error.

© 1997-2013 Devart. All Rights Reserved.

**17.10.1.1.2.2 ErrorCode Property**

Determines the error code returned by the server.

**Class**[EDAEError](#)**Syntax**

```
property ErrorCode: integer;
```

**Remarks**

Use the ErrorCode property to determine the error code returned by InterBase. This value is always positive.

**See Also**

- [EIBCEError.ErrorNumber](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.2 TCRDataSource Class

Provides an interface between a DAC dataset components and data-aware controls on a form.

For a list of all members of this type, see [TCRDataSource](#) members.

#### Unit

[DBAccess](#)

#### Syntax

```
TCRDataSource = class (TDataSource);
```

#### Remarks

TCRDataSource provides an interface between a DAC dataset components and data-aware controls on a form.

TCRDataSource inherits its functionality directly from the TDataSource component. At design time assign individual data-aware components' DataSource properties from their drop-down listboxes.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.2.1 Members

[TCRDataSource](#) class overview.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.3 TCustomConnectDialog Class

A base class for the connect dialog components.

For a list of all members of this type, see [TCustomConnectDialog](#) members.

#### Unit

[DBAccess](#)

#### Syntax

```
TCustomConnectDialog = class (TComponent);
```

#### Remarks

TCustomConnectDialog is a base class for the connect dialog components. It provides functionality to show a dialog box where user can edit username, password and server name before connecting to a database. You can customize captions of buttons and labels by their properties.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.3.1 Members

[TCustomConnectDialog](#) class overview.

#### Properties

Name	Description
------	-------------

---

<a href="#">CancelButton</a>	Used to specify the label for the Cancel button.
<a href="#">Caption</a>	Used to set the caption of dialog box.
<a href="#">ConnectButton</a>	Used to specify the label for the Connect button.
<a href="#">DialogClass</a>	Used to specify the class of the form that will be displayed to enter login information.
<a href="#">LabelSet</a>	Used to set the language of buttons and labels captions.
<a href="#">PasswordLabel</a>	Used to specify a prompt for password edit.
<a href="#">Retries</a>	Used to indicate the number of retries of failed connections.
<a href="#">SavePassword</a>	Used for the password to be displayed in ConnectDialog in asterisks.
<a href="#">ServerLabel</a>	Used to specify a prompt for the server name edit.
<a href="#">StoreLogInfo</a>	Used to specify whether the login information should be kept in system registry after a connection was established.
<a href="#">UsernameLabel</a>	Used to specify a prompt for username edit.

## Methods

Name	Description
<a href="#">Execute</a>	Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button.
<a href="#">GetServerList</a>	Retrieves a list of available server names.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.3.2 Properties

Properties of the **TCustomConnectDialog** class.  
For a complete list of the **TCustomConnectDialog** class members, see the [TCustomConnectDialog Members](#) topic.

## Public

Name	Description
------	-------------

[CancelButton](#)

Used to specify the label for the Cancel button.

[Caption](#)

Used to set the caption of dialog box.

[ConnectButton](#)

Used to specify the label for the Connect button.

[DialogClass](#)

Used to specify the class of the form that will be displayed to enter login information.

[LabelSet](#)

Used to set the language of buttons and labels captions.

[PasswordLabel](#)

Used to specify a prompt for password edit.

[Retries](#)

Used to indicate the number of retries of failed connections.

[SavePassword](#)

Used for the password to be displayed in ConnectDialog in asterisks.

[ServerLabel](#)

Used to specify a prompt for the server name edit.

[StoreLogInfo](#)

Used to specify whether the login information should be kept in system registry after a connection was established.

[UsernameLabel](#)

Used to specify a prompt for username edit.

### See Also

- [TCustomConnectDialog Class](#)
- [TCustomConnectDialog Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.3.2.1 CancelButton Property

Used to specify the label for the Cancel button.

### Class

[TCustomConnectDialog](#)

### Syntax

```
property CancelButton: string;
```

### Remarks

Use the CancelButton property to specify the label for the Cancel button.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.3.2.2 Caption Property

Used to set the caption of dialog box.

##### **Class**

[TCustomConnectDialog](#)

##### **Syntax**

```
property Caption: string;
```

##### **Remarks**

Use the Caption property to set the caption of dialog box.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.3.2.3 ConnectButton Property

Used to specify the label for the Connect button.

##### **Class**

[TCustomConnectDialog](#)

##### **Syntax**

```
property ConnectButton: string;
```

##### **Remarks**

Use the ConnectButton property to specify the label for the Connect button.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.3.2.4 DialogClass Property

Used to specify the class of the form that will be displayed to enter login information.

##### **Class**

[TCustomConnectDialog](#)

##### **Syntax**

```
property DialogClass: string;
```

##### **Remarks**

Use the DialogClass property to specify the class of the form that will be displayed to enter login information. When this property is blank, TCustomConnectDialog uses the default form - TConnectForm. You can write your own login form to enter login information and assign its class name to the DialogClass property. Each login form must have TCustomConnectDialog published property to access connection information. For details see the implementation of the connect form which sources are in the Lib subdirectory of the IBDAC installation directory.

## See Also

- [GetServerList](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.3.2.5 LabelSet Property

Used to set the language of buttons and labels captions.

#### Class

[TCustomConnectDialog](#)

#### Syntax

```
property LabelSet: TLabelSet default lsEnglish;
```

#### Remarks

Use the LabelSet property to set the language of labels and buttons captions.  
The default value is lsEnglish.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.3.2.6 PasswordLabel Property

Used to specify a prompt for password edit.

#### Class

[TCustomConnectDialog](#)

#### Syntax

```
property PasswordLabel: string;
```

#### Remarks

Use the PasswordLabel property to specify a prompt for password edit.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.3.2.7 Retries Property

Used to indicate the number of retries of failed connections.

#### Class

[TCustomConnectDialog](#)

#### Syntax

```
property Retries: word default 3;
```

#### Remarks

Use the Retries property to determine the number of retries of failed connections.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.3.2.8 SavePassword Property

Used for the password to be displayed in ConnectDialog in asterisks.

**Class**

[TCustomConnectDialog](#)

**Syntax**

```
property SavePassword: boolean default False;
```

**Remarks**

If True, and the Password property of the connection instance is assigned, the password in ConnectDialog is displayed in asterisks.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.3.2.9 ServerLabel Property

Used to specify a prompt for the server name edit.

**Class**

[TCustomConnectDialog](#)

**Syntax**

```
property ServerLabel: string;
```

**Remarks**

Use the ServerLabel property to specify a prompt for the server name edit.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.3.2.10 StoreLogInfo Property

Used to specify whether the login information should be kept in system registry after a connection was established.

**Class**

[TCustomConnectDialog](#)

**Syntax**

```
property StoreLogInfo: boolean default True;
```

**Remarks**

Use the StoreLogInfo property to specify whether to keep login information in system registry after a connection was established using provided username, password and servername.  
Set this property to True to store login information.  
The default value is True.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.3.2.11 UsernameLabel Property

Used to specify a prompt for username edit.

**Class**

[TCustomConnectDialog](#)

**Syntax**

```
property UsernameLabel: string;
```

**Remarks**

Use the UsernameLabel property to specify a prompt for username edit.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.3.3 Methods

Methods of the **TCustomConnectDialog** class.

For a complete list of the **TCustomConnectDialog** class members, see the [TCustomConnectDialog Members](#) topic.

**Public**

Name	Description
<a href="#">Execute</a>	Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button.
<a href="#">GetServerList</a>	Retrieves a list of available server names.

**See Also**

- [TCustomConnectDialog Class](#)
  - [TCustomConnectDialog Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.3.3.1 Execute Method

Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button.

**Class**

[TCustomConnectDialog](#)

**Syntax**

```
function Execute: boolean; virtual;
```

**Return Value**

True, if connected.

**Remarks**

Displays the connect dialog and calls the connection's Connect method when user



clicks the Connect button. Returns True if connected. If user clicks Cancel, Execute returns False.

In the case of failed connection Execute offers to connect repeat [Retries](#) times.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.3.3.2 GetServerList Method

Retrieves a list of available server names.

### Class

[TCustomConnectDialog](#)

### Syntax

```
procedure GetServerList(List: _TStrings); virtual;
```

### Parameters

*List*

Holds a list of available server names.

### Remarks

Call the GetServerList method to retrieve a list of available server names. It is particularly relevant for writing custom login form.

### See Also

- [DialogClass](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4 TCustomDAConnection Class

A base class for components used to establish connections.

For a list of all members of this type, see [TCustomDAConnection](#) members.

### Unit

[DBAccess](#)

### Syntax

```
TCustomDAConnection = class (TCustomConnection);
```

### Remarks

TCustomDAConnection is a base class for components that establish connection with database, provide customised login support, and perform transaction control. Do not create instances of TCustomDAConnection. To add a component that represents a connection to a source of data, use descendants of the TCustomDAConnection class.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.4.1 Members

[TCustomDAConnection](#) class overview.

**Properties**

Name	Description
<a href="#">ConnectDialog</a>	Allows to link a <a href="#">TCustomConnectDialog</a> component.
<a href="#">ConvertEOL</a>	Allows customi ing line breaks in string fields and parameters.
<a href="#">InTransaction</a>	Indicates whether the transaction is active.
<a href="#">LoginPrompt</a>	Specifies whether a login dialog appears immediately before opening a new connection.
<a href="#">Options</a>	Specifies the connection behavior.
<a href="#">Password</a>	Serves to supply a password for login.
<a href="#">Pooling</a>	Enables or disables using connection pool.
<a href="#">PoolingOptions</a>	Specifies the behaviour of connection pool.
<a href="#">Server</a>	Serves to supply the server name for login.
<a href="#">Username</a>	Used to supply a user name for login.

**Methods**

Name	Description
<a href="#">ApplyUpdates</a>	Overloaded. Applies changes in datasets.
<a href="#">Commit</a>	Commits current transaction.
<a href="#">Connect</a>	Establishes a connection to the server.
<a href="#">CreateDataSet</a>	Creates a dataset component.
<a href="#">CreateSQL</a>	Creates a component for queries execution.
<a href="#">Disconnect</a>	Performs disconnect.
<a href="#">ExecProc</a>	Allows to execute stored procedure or function providing its name and parameters.

<a href="#">ExecProcEx</a>	Allows to execute a stored procedure or function.
<a href="#">ExecSQL</a>	Executes a SQL statement with parameters.
<a href="#">ExecSQLEx</a>	Executes any SQL statement outside the TQuery or TSQL components.
<a href="#">GetDatabaseNames</a>	Returns a database list from the server.
<a href="#">GetStoredProcNames</a>	Returns a list of stored procedures from the server.
<a href="#">GetTableNames</a>	Provides a list of available tables names.
<a href="#">MonitorMessage</a>	Sends a specified message through the <a href="#">TCustomDASQLMonitor</a> component.
<a href="#">RemoveFromPool</a>	Marks the connection that should not be returned to the pool after disconnect.
<a href="#">Rollback</a>	Discards all current data changes and ends transaction.
<a href="#">StartTransaction</a>	Begins a new user transaction.

## Events

Name	Description
<a href="#">OnConnectionLost</a>	This event occurs when connection was lost.
<a href="#">OnError</a>	This event occurs when an error has arisen in the connection.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.4.2 Properties

Properties of the **TCustomDAConnection** class.  
For a complete list of the **TCustomDAConnection** class members, see the [TCustomDAConnection Members](#) topic.

## Public

Name	Description
<a href="#">ConnectDialog</a>	Allows to link a <a href="#">TCustomConnectDialog</a> component.

[ConvertEOL](#)

Allows customizing line breaks in string fields and parameters.

[InTransaction](#)

Indicates whether the transaction is active.

[LoginPrompt](#)

Specifies whether a login dialog appears immediately before opening a new connection.

[Options](#)

Specifies the connection behavior.

[Password](#)

Serves to supply a password for login.

[Pooling](#)

Enables or disables using connection pool.

[PoolingOptions](#)

Specifies the behaviour of connection pool.

[Server](#)

Serves to supply the server name for login.

[Username](#)

Used to supply a user name for login.

**See Also**

- [TCustomDAConnection Class](#)
  - [TCustomDAConnection Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.2.1 ConnectDialog Property

Allows to link a [TCustomConnectDialog](#) component.

**Class**[TCustomDAConnection](#)**Syntax**

**property** ConnectDialog: [TCustomConnectDialog](#);

**Remarks**

Use the ConnectDialog property to assign to connection a [TCustomConnectDialog](#) component.

**See Also**

- [TCustomConnectDialog](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.4.2.2 ConvertEOL Property

Allows customizing line breaks in string fields and parameters.

**Class**

[TCustomDAConnection](#)

**Syntax**

```
property ConvertEOL: boolean default False;
```

**Remarks**

Affects the line break behavior in string fields and parameters. When fetching strings (including the text BLOB fields) with ConvertEOL = True, dataset converts their line breaks from the LF to CRLF form. And when posting strings to server with ConvertEOL turned on, their line breaks are converted from CRLF to LF form. By default, strings are not converted.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.4.2.3 InTransaction Property

Indicates whether the transaction is active.

**Class**

[TCustomDAConnection](#)

**Syntax**

```
property InTransaction: boolean;
```

**Remarks**

Examine the InTransaction property at runtime to determine whether user transaction is currently in progress. In other words InTransaction is set to True when user explicitly calls [StartTransaction](#). Calling [Commit](#) or [Rollback](#) sets InTransaction to False. The value of the InTransaction property cannot be changed directly.

**See Also**

- [StartTransaction](#)
- [Commit](#)
- [Rollback](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.4.2.4 LoginPrompt Property

Specifies whether a login dialog appears immediately before opening a new connection.

**Class**

[TCustomDAConnection](#)

**Syntax**

```
property LoginPrompt default True;
```

**Remarks**

Specifies whether a login dialog appears immediately before opening a new connection. If [ConnectDialog](#) is not specified, the default connect dialog will be shown. The connect dialog will appear only if the `IbDacVcl` unit appears to the uses clause.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.4.2.5 Options Property

Specifies the connection behavior.

**Class**

[TCustomDAConnection](#)

**Syntax**

```
property Options: TDACConnectionOptions;
```

**Remarks**

Set the properties of Options to specify the behaviour of the connection. Descriptions of all options are in the table below.

Option Name	Description
<a href="#">DefaultSortType</a>	Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the <a href="#">TMemDataSet.IndexFieldNames</a> property of a dataset.
<a href="#">DisconnectedMode</a>	Used to open a connection only when needed for performing a server call and closes after performing the operation.
<a href="#">KeepDesignConnected</a>	Used to prevent an application from establishing a connection at the time of startup.
<a href="#">LocalFailover</a>	If True, the <a href="#">OnConnectionLost</a> event occurs and a failover operation can be performed after connection breaks.

**See Also**

- [Disconnected Mode](#)
  - [Working in an Unstable Network](#)
-

©

1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.2.6 Password Property

Serves to supply a password for login.

### Class

[TCustomDAConnection](#)

### Syntax

```
property Password: string;
```

### Remarks

Use the Password property to supply a password to handle server's request for a login.

**Warning:** Storing hard-coded user name and password entries as property values or in code for the OnLogin event handler can compromise server security.

### See Also

- [Username](#)
- [Server](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.2.7 Pooling Property

Enables or disables using connection pool.

### Class

[TCustomDAConnection](#)

### Syntax

```
property Pooling: boolean default False;
```

### Remarks

Normally, when TCustomDAConnection establishes connection with the server it takes server memory and time resources for allocating new server connection. For example, pooling can be very useful when using disconnect mode. If an application has wide user activity that forces many connect/disconnect operations, it may spend a lot of time on creating connection and sending requests to the server. TCustomDAConnection has software pool which stores open connections with identical parameters.

Connection pool uses separate thread that validates the pool every 30 seconds. Pool validation consists of checking each connection in the pool. If a connection is broken due to a network problem or another reason, it is deleted from the pool. The validation procedure removes also connections that are not used for a long time

even if they are valid from the pool.

Set Pooling to True to enable pooling. Specify correct values for PoolingOptions. Two connections belong to the same pool if they have identical values for the parameters: [MinPoolSize](#), [MaxPoolSize](#), [Validate](#), [ConnectionLifetime](#), [TIBCCConnection.Database](#), [TIBCCConnectionOptions.Charset](#), [TIBCCConnectionOptions.UseUnicode](#), [TIBCCConnectionOptions.Role](#), [TIBCCConnection.SQLDialect](#), [TIBCCConnection.Params](#).

**Note:** Using Pooling := True can cause errors with working with temporary tables.

## See Also

- [Username](#)
- [Password](#)
- [PoolingOptions](#)
- [Connection Pooling](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.4.2.8 PoolingOptions Property

Specifies the behaviour of connection pool.

## Class

[TCustomDAConnection](#)

## Syntax

**property** PoolingOptions: [TPoolingOptions](#);

## Remarks

Set the properties of PoolingOptions to specify the behaviour of connection pool. Descriptions of all options are in the table below.

Option Name	Description
<a href="#">ConnectionLifetime</a>	Used to specify the maximum time during which an opened connection can be used by connection pool.
<a href="#">MaxPoolSize</a>	Used to specify the maximum number of connections that can be opened in connection pool.
<a href="#">MinPoolSize</a>	Used to specify the minimum number of connections that can be opened in the connection pool.
<a href="#">Validate</a>	Used for a connection to be validated when it is returned from the pool.

## See Also

- [Pooling](#)



©

1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.2.9 Server Property

Serves to supply the server name for login.

### Class

[TCustomDAConnection](#)

### Syntax

```
property Server: string;
```

### Remarks

Use the Server property to supply server name to handle server's request for a login.

### See Also

- [Username](#)
- [Password](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.2.10 Username Property

Used to supply a user name for login.

### Class

[TCustomDAConnection](#)

### Syntax

```
property Username: string;
```

### Remarks

Use the Username property to supply a user name to handle server's request for login. If this property is not set, IBDAC tries to connect with the user name.

**Warning:** Storing hard-coded user name and password entries as property values or in code for the OnLogin event handler can compromise server security.

### See Also

- [Password](#)
- [Server](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.4.3 Methods

Methods of the **TCustomDAConnection** class.

For a complete list of the **TCustomDAConnection** class members, see the [TCustomDAConnection Members](#) topic.

**Public**

Name	Description
<a href="#">ApplyUpdates</a>	Overloaded. Applies changes in datasets.
<a href="#">Commit</a>	Commits current transaction.
<a href="#">Connect</a>	Establishes a connection to the server.
<a href="#">CreateDataSet</a>	Creates a dataset component.
<a href="#">CreateSQL</a>	Creates a component for queries execution.
<a href="#">Disconnect</a>	Performs disconnect.
<a href="#">ExecProc</a>	Allows to execute stored procedure or function providing its name and parameters.
<a href="#">ExecProcEx</a>	Allows to execute a stored procedure or function.
<a href="#">ExecSQL</a>	Executes a SQL statement with parameters.
<a href="#">ExecSQLEx</a>	Executes any SQL statement outside the TQuery or TSQL components.
<a href="#">GetDatabaseNames</a>	Returns a database list from the server.
<a href="#">GetStoredProcNames</a>	Returns a list of stored procedures from the server.
<a href="#">GetTableNames</a>	Provides a list of available tables names.
<a href="#">MonitorMessage</a>	Sends a specified message through the <a href="#">TCustomDASQLMonitor</a> component.
<a href="#">RemoveFromPool</a>	Marks the connection that should not be returned to the pool after disconnect.
<a href="#">Rollback</a>	Discards all current data changes and ends transaction.
<a href="#">StartTransaction</a>	Begins a new user transaction.

**See Also**

- [TCustomDAConnection Class](#)
- [TCustomDAConnection Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.3.1 ApplyUpdates Method

Applies changes in datasets.

**Class**

[TCustomDAConnection](#)

**Overload List**

Name	Description
<a href="#">ApplyUpdates</a>	Applies changes from all active datasets.
<a href="#">ApplyUpdates(DataSets: <b>array of TCustomDADataSet</b>)</a>	Applies changes from the specified datasets.

© 1997-2013 Devart. All Rights Reserved.

Applies changes from all active datasets.

**Class**

[TCustomDAConnection](#)

**Syntax**

```
procedure ApplyUpdates; overload; virtual
```

**Remarks**

Call the ApplyUpdates method to write all pending cached updates from all active datasets attached to this connection to a database or from specific datasets. The ApplyUpdates method passes cached data to the database for storage, takes care of committing or rolling back transactions, and clearing the cache when the operation is successful.

Using ApplyUpdates for connection is a preferred method of updating datasets rather than calling each individual dataset's ApplyUpdates method.

**See Also**

- [TMemDataSet.CachedUpdates](#)
- [TMemDataSet.ApplyUpdates](#)

© 1997-2013 Devart. All Rights Reserved.

Applies changes from the specified datasets.

**Class**

[TCustomDAConnection](#)

### Syntax

```
procedure ApplyUpdates(DataSets: array of TCustomDADataSet);  
overload; virtual
```

### Parameters

*DataSets*

A list of datasets changes in which are to be applied.

### Remarks

Call the ApplyUpdates method to write all pending cached updates from the specified datasets. The ApplyUpdates method passes cached data to the database for storage, takes care of committing or rolling back transactions and clearing the cache when operation is successful.

Using ApplyUpdates for connection is a preferred method of updating datasets rather than calling each individual dataset's ApplyUpdates method.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.3.2 Commit Method

Commits current transaction.

### Class

[TCustomDAConnection](#)

### Syntax

```
procedure Commit; virtual;
```

### Remarks

Call the Commit method to commit current transaction. On commit server writes permanently all pending data updates associated with the current transaction to the database and then ends the transaction. The current transaction is the last transaction started by calling StartTransaction.

### See Also

- [Rollback](#)
  - [StartTransaction](#)
  - P:Devart.IbDac.TCustomIBCDDataSet.FetchAll
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.3.3 Connect Method

Establishes a connection to the server.

### Class

[TCustomDAConnection](#)

## Syntax

```
procedure Connect;
```

## Remarks

Call the Connect method to establish a connection to the server. Connect sets the Connected property to True. If LoginPrompt is True, Connect prompts user for login information as required by the server, or otherwise tries to establish a connection using values provided in the [Username](#), [Password](#), and [Server](#) properties.

## See Also

- [Disconnect](#)
- [Username](#)
- [Password](#)
- [Server](#)
- [ConnectDialog](#)
- [TIBCCConnection.Connected](#)
- [TIBCCConnection.AssignConnect](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.4.3.4 CreateDataSet Method

Creates a dataset component.

## Class

[TCustomDAConnection](#)

## Syntax

```
function CreateDataSet: TCustomDADataset; virtual;
```

## Return Value

Returns a new instance of the class.

## Remarks

Call the CreateDataSet method to return a new instance of the [TCustomDADataset](#) class and associate it with this connection object. In the descendant classes this method should be overridden to create an appropriate descendant of the TCustomDADataset component.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.4.3.5 CreateSQL Method

Creates a component for queries execution.

## Class

[TCustomDAConnection](#)

## Syntax

```
function CreateSQL: TCustomDASQL; virtual;  
Return Value
```

A new instance of the class.

### Remarks

Call the CreateSQL to return a new instance of the [TCustomDASQL](#) class and associates it with this connection object. In the descendant classes this method should be overridden to create an appropriate descendant of the TCustomDASQL component.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.3.6 Disconnect Method

Performs disconnect.

### Class

[TCustomDAConnection](#)

### Syntax

```
procedure Disconnect;
```

### Remarks

Call the Disconnect method to drop a connection to database. Before the connection component is deactivated, all associated datasets are closed. Calling Disconnect is similar to setting the Connected property to False.

In most cases, closing a connection frees system resources allocated to the connection.

If user transaction is active, e.g. the [InTransaction](#) flag is set, calling to Disconnect will lead to applying the action specified in TIBCTransaction.DefaultCloseAction for the current user transaction.

**Note:** If a previously active connection is closed and then reopened, any associated datasets must be individually reopened; reopening the connection does not automatically reopen associated datasets.

### See Also

- [Connect](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.3.7 ExecProc Method

Allows to execute stored procedure or function providing its name and parameters.

### Class

[TCustomDAConnection](#)

### Syntax

```
function ExecProc (Name: string; const Params: array of variant):
```

---

```
variant; virtual;
```

### Parameters

#### *Name*

Holds the name of the stored procedure or function.

#### *Params*

Holds the parameters of the stored procedure or function.

### Return Value

the result of the stored procedure.

## Remarks

Allows to execute stored procedure or function providing its name and parameters. Use the following Name value syntax for executing specific overloaded routine: "StoredProcName:1" or "StoredProcName:5". The first example executes the first overloaded stored procedure, while the second example executes the fifth overloaded procedure.

Assign parameters' values to the Params array in exactly the same order and number as they appear in the stored procedure declaration. Out parameters of the procedure can be accessed with the ParamByName procedure.

If the value of an input parameter was not included to the Params array, parameter default value is taken. Only parameters at the end of the list can be unincluded to the Params array. If the parameter has no default value, the NULL value is sent.

**Note:** Stored functions unlike stored procedures return result values that are obtained internally through the RESULT parameter. You will no longer have to provide anonymous value in the Params array to describe the result of the function. The stored function result is obtained from the Params[0 indexed property or with the ParamByName('RESULT') method call.

For further examples of parameter usage see [ExecSQL](#), [ExecSQLEx](#).

## Example

For example, having stored function declaration presented in Example 1), you may execute it and retrieve its result with commands presented in Example 2):

Example 1)

```
CREATE procedure MY SUM (  
    A INTEGER,  
    B INTEGER)  
RETURNS (  
    RESULT INTEGER)  
  
as  
begin  
    Result = a + b;  
end;
```

Example 2)

```
Label1.Caption:= MyIBConnection1.ExecProc('My Sum', [10, 20]);  
Label2.Caption:= MyIBConnection1.ParamByName('Result').AsString;
```

## See Also

-

[ExecProcEx](#)

- [ExecSQL](#)
  - [ExecSQLEx](#)
  - [TIBCCConnection.ParamByName](#)
  - [TIBCCConnection.SQL](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.3.8 ExecProcEx Method

Allows to execute a stored procedure or function.

### Class

[TCustomDAConnection](#)

### Syntax

```
function ExecProcEx(Name: string; const Params: array of variant):  
variant; virtual;
```

#### Parameters

##### *Name*

Holds the stored procedure name.

##### *Params*

Holds an array of pairs of parameters' names and values.

#### Return Value

the result of the stored procedure.

### Remarks

Allows to execute a stored procedure or function. Provide the stored procedure name and its parameters to the call of ExecProcEx.

Use the following Name value syntax for executing specific overloaded routine: "StoredProcName:1" or "StoredProcName:5". The first example executes the first overloaded stored procedure, while the second example executes the fifth overloaded procedure.

Assign pairs of parameters' names and values to a Params array so that every name comes before its corresponding value when an array is being indexed.

Out parameters of the procedure can be accessed with the ParamByName procedure. If the value for an input parameter was not included to the Params array, the parameter default value is taken. If the parameter has no default value, the NULL value is sent.

**Note:** Stored functions unlike stored procedures return result values that are obtained internally through the RESULT parameter. You will no longer have to provide anonymous value in the Params array to describe the result of the function. Stored function result is obtained from the Params[0 indexed property or with the ParamByName('RESULT') method call.

For an example of parameters usage see [ExecSQLEx](#).

### Example

If you have some stored procedure accepting four parameters, and you want to



```
Connection.ExecProcEx('Some_Stored_Procedure', ['Param_Name1', 'Param_Value1', ''])
```

- ExecSQL
- ExecSQLEx
- ExecProc

#### 17.10.1.4.3.9 ExecSQL Method

© 2013 Enter your company name

## See Also

- [ExecSQLEx](#)
  - [ExecProc](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.4.3.10 ExecSQLEx Method

Executes any SQL statement outside the TQuery or TSQL components.

## Class

[TCustomDAConnection](#)

## Syntax

```
function ExecSQLEx(Text: string; const Params: array of variant):  
variant; virtual;
```

### Parameters

#### *Text*

a SQL statement to be executed.

#### *Params*

Array of parameter values arranged in the same order as they appear in SQL statement.

### Return Value

Out parameter with the name Result will hold the result of a function having data type dtString. Otherwise returns Null.

## Remarks

Call the ExecSQLEx method to execute any SQL statement outside the TQuery or TSQL components. Supply the Params array with values arranged in pairs of parameter name and its value. This way each parameter name in the array is found on even index values whereas parameter value is on odd index value but right after its parameter name. The parameter pairs must be arranged according to their occurrence in a SQL statement which itself is passed in the Text string parameter. The Params array must contain all IN and OUT parameters defined in the SQL statement. For OUT parameters provide any values of valid types so that they are explicitly defined before call to the ExecSQLEx method.

Out parameter with the name Result will hold the result of a function having data type dtString. If neither of the parameters in the Text statement is named Result, ExecSQLEx will return Null.

To get the values of OUT parameters use the ParamByName function.

## Example

```
IBCCConnection.ExecSQLEx('begin :A:= :B + :C; end;',  
  ['A', 0, 'B', 5, 'C', 3]);  
A:= IBCCConnection.ParamByName('A').AsInteger;
```

## See Also

- [ExecSQL](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.4.3.11 GetDatabaseNames Method

Returns a database list from the server.

## Class

[TCustomDAConnection](#)

## Syntax

```
procedure GetDatabaseNames(List: _TStrings); virtual;
```

### Parameters

*List*

A TStrings descendant that will be filled with database names.

## Remarks

Populates a string list with the names of databases.

**Note:** Any contents already in the target string list object are eliminated and overwritten by data produced by GetDatabaseNames.

## See Also

- [GetTableNames](#)
- [GetStoredProcNames](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.4.3.12 GetStoredProcNames Method

Returns a list of stored procedures from the server.

## Class

[TCustomDAConnection](#)

## Syntax

```
procedure GetStoredProcNames(List: _TStrings; AllProcs: boolean =  
False); virtual;
```

### Parameters

*List*

A TStrings descendant that will be filled with the names of stored procedures in the database.

*AllProcs*

True, if stored procedures from all schemas or including system procedures

(depending on the server) are returned. False otherwise.

### Remarks

Call the `GetStoredProcNames` method to get the names of available stored procedures and functions. `GetStoredProcNames` populates a string list with the names of stored procs in the database. If `AllProcs = True`, the procedure returns to the `List` parameter the names of the stored procedures that belong to all schemas; otherwise, `List` will contain the names of functions that belong to the current schema.

**Note:** Any contents already in the target string list object are eliminated and overwritten by data produced by `GetStoredProcNames`.

### See Also

- [GetDatabaseNames](#)
  - [GetTableNames](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.3.13 GetTableNames Method

Provides a list of available tables names.

### Class

[TCustomDAConnection](#)

### Syntax

```
procedure GetTableNames(List: _TStrings; AllTables: boolean =  
False; OnlyTables: boolean = False); virtual;
```

#### Parameters

##### *List*

A `TStrings` descendant that will be filled with table names.

##### *AllTables*

True, if procedure returns all table names including the names of system tables to the `List` parameter.

##### *OnlyTables*

### Remarks

Call the `GetTableNames` method to get the names of available tables. Populates a string list with the names of tables in the database. If `AllTables = True`, procedure returns all table names including the names of system tables to the `List` parameter, otherwise `List` will not contain the names of system tables. If `AllTables = True`, the procedure returns to the `List` parameter the names of the tables that belong to all schemas; otherwise, `List` will contain the names of the tables that belong to the current schema.

**Note:** Any contents already in the target string list object are eliminated and overwritten by the data produced by `GetTableNames`.

## See Also

- [GetDatabaseNames](#)
- [GetStoredProcNames](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.4.3.14 MonitorMessage Method

Sends a specified message through the [TCustomDASQLMonitor](#) component.

## Class

[TCustomDAConnection](#)

## Syntax

```
procedure MonitorMessage(const Msg: string);
```

### Parameters

*Msg*

Message text that will be sent.

## Remarks

Call the MonitorMessage method to output specified message via the [TCustomDASQLMonitor](#) component.

## See Also

- [TCustomDASQLMonitor](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.4.3.15 RemoveFromPool Method

Marks the connection that should not be returned to the pool after disconnect.

## Class

[TCustomDAConnection](#)

## Syntax

```
procedure RemoveFromPool;
```

## Remarks

Call the RemoveFromPool method to mark the connection that should be deleted after disconnect instead of returning to the connection pool.

## See Also

- [Pooling](#)
- [PoolingOptions](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.3.16 Rollback Method

Discards all current data changes and ends transaction.

### Class

[TCustomDAConnection](#)

### Syntax

```
procedure Rollback; virtual;
```

### Remarks

Call the Rollback method to discard all updates, insertions, and deletions of data associated with the current transaction to the database server and then end the transaction. The current transaction is the last transaction started by calling [StartTransaction](#).

### See Also

- [Commit](#)
  - [StartTransaction](#)
  - P:Devart.IbDac.TCustomIBCDDataSet.FetchAll
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.4.3.17 StartTransaction Method

Begins a new user transaction.

### Class

[TCustomDAConnection](#)

### Syntax

```
procedure StartTransaction; virtual;
```

### Remarks

Call the StartTransaction method to begin a new user transaction against the database server. Before calling StartTransaction, an application should check the status of the [InTransaction](#) property. If InTransaction is True, indicating that a transaction is already in progress, a subsequent call to StartTransaction without first calling [Commit](#) or [Rollback](#) to end the current transaction raises EDatabaseError. Calling StartTransaction when connection is closed also raises EDatabaseError.

Updates, insertions, and deletions that take place after a call to StartTransaction are held by the server until an application calls Commit to save the changes, or Rollback to cancel them.

## See Also

- [Commit](#)
- [Rollback](#)
- [InTransaction](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.4.4 Events

Events of the **TCustomDAConnection** class.

For a complete list of the **TCustomDAConnection** class members, see the [TCustomDAConnection Members](#) topic.

## Public

Name	Description
<a href="#">OnConnectionLost</a>	This event occurs when connection was lost.
<a href="#">OnError</a>	This event occurs when an error has arisen in the connection.

## See Also

- [TCustomDAConnection Class](#)
- [TCustomDAConnection Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.4.4.1 OnConnectionLost Event

This event occurs when connection was lost.

## Class

[TCustomDAConnection](#)

## Syntax

```
property OnConnectionLost: TConnectionLostEvent;
```

## Remarks

Write the OnConnectionLost event handler to process fatal errors and perform failover.

**Note:** you should explicitly add the [MemData](#) unit to the 'uses' list to use the OnConnectionLost event handler.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.4.4.2 OnError Event

This event occurs when an error has arisen in the connection.

## Class

### [TCustomDAConnection](#)

#### Syntax

**property** OnError: [TDAConnectionErrorEvent](#);

#### Remarks

Write the OnError event handler to respond to errors that arise with connection. Check the E parameter to get the error code. Set the Fail parameter to False to prevent an error dialog from being displayed and to raise the EAbort exception to cancel current operation. The default value of Fail is True.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5 TCustomDADataset Class

Encapsulates general set of properties, events, and methods for working with data accessed through various database engines.

For a list of all members of this type, see [TCustomDADataset](#) members.

#### Unit

[DBAccess](#)

#### Syntax

TCustomDADataset = **class** ([TMemDataSet](#)) ;

#### Remarks

TCustomDADataset encapsulates general set of properties, events, and methods for working with data accessed through various database engines. All database-specific features are supported by descendants of TCustomDADataset.

Applications should not use TCustomDADataset objects directly.

#### Inheritance Hierarchy

[TMemDataSet](#)

**TCustomDADataset**

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.1 Members

[TCustomDADataset](#) class overview.

#### Properties

Name	Description
<a href="#">BaseSQL</a>	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.



<a href="#">Connection</a>	Used to specify a connection object to use to connect to a data store.
<a href="#">Debug</a>	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#">DetailFields</a>	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#">Disconnected</a>	Used to keep dataset opened after connection is closed.
<a href="#">Encryption</a>	Used to specify the options of the data encryption in a dataset.
<a href="#">FetchRows</a>	Used to define the number of rows to be transferred across the network at the same time.
<a href="#">FilterSQL</a>	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#">FinalSQL</a>	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">IsQuery</a>	Used to check whether SQL statement returns rows.
<a href="#">KeyFields</a>	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.

[MacroCount](#)

Used to get the number of macros associated with the Macros property.

[Macros](#)

Makes it possible to change SQL queries easily.

[MasterFields](#)

Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.

[MasterSource](#)

Used to specify the data source component which binds current dataset to the master one.

[Options](#)

Used to specify the behaviour of TCustomDADataset object.

[ParamCheck](#)

Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.

[ParamCount](#)

Used to indicate how many parameters are there in the Params property.

[Params](#)

Used to view and set parameter names, values, and data types dynamically.

[Prepared](#) (inherited from [TMemDataSet](#))

Determines whether a query is prepared for execution or not.

[ReadOnly](#)

Used to prevent users from updating, inserting, or deleting data in the dataset.

[RefreshOptions](#)

Used to indicate when the editing record is refreshed.

[RowsAffected](#)

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

[SQL](#)

Used to provide a SQL statement that a query component executes when its Open method is called.

[SQLDelete](#)

Used to specify a SQL statement that will be used when applying a deletion to a record.

<a href="#">SQLInsert</a>	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#">SQLLock</a>	Used to specify a SQL statement that will be used to perform a record lock.
<a href="#">SQLRefresh</a>	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#">TCustomDADataset.RefreshRecord</a> procedure.
<a href="#">SQLUpdate</a>	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#">UniDirectional</a>	Used if an application does not need bidirectional access to records in the result set.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.

## Methods

Name	Description
<a href="#">AddWhere</a>	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">BreakExec</a>	Breaks execution of the SQL statement on the server.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">CreateBlobStream</a>	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.

<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">DeleteWhere</a>	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#">Execute</a>	Executes a SQL statement on the server.
<a href="#">Executing</a>	Indicates whether SQL statement is still being executed.
<a href="#">Fetched</a>	Used to learn whether TCustomDADataset has already fetched all rows.
<a href="#">Fetching</a>	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#">FetchingAll</a>	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#">FindKey</a>	Searches for a record which contains specified field values.
<a href="#">FindMacro</a>	Indicates whether a specified macro exists in a dataset.
<a href="#">FindNearest</a>	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#">FindParam</a>	Determines if a parameter with the specified name exists in a dataset.
<a href="#">GetBlob</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">GetDataType</a>	Returns internal field types defined in the MemData and accompanying modules.
<a href="#">GetFieldObject</a>	Returns a multireference shared object from field.
<a href="#">GetFieldPrecision</a>	Retrieves the precision of a number field.
<a href="#">GetFieldScale</a>	Retrieves the scale of a number field.

<a href="#">GetOrderBy</a>	Retrieves an ORDER BY clause from a SQL statement.
<a href="#">GotoCurrent</a>	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">Lock</a>	Locks the current record.
<a href="#">MacroByName</a>	Finds a Macro with the name passed in Name.
<a href="#">ParamByName</a>	Sets or uses parameter information for a specific parameter based on its name.
<a href="#">Prepare</a>	Allocates, opens, and parses cursor for a query.
<a href="#">RefreshRecord</a>	Actuali es field values for the current record.
<a href="#">RestoreSQL</a>	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a>	Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveSQL</a>	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

<a href="#">SetOrderBy</a>	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQLSaved</a>	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">UnLock</a>	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the <a href="#">ApplyUpdates</a> method while cached updates are enabled.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.

## Events

Name	Description
<a href="#">AfterExecute</a>	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a>	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a>	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">BeforeFetch</a>	Occurs before dataset is going to fetch block of records from the server.
<a href="#">BeforeUpdateExecute</a>	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.2 Properties

Properties of the **TCustomDADataset** class.  
For a complete list of the **TCustomDADataset** class members, see the [TCustomDADataset Members](#) topic.

**Public**

Name	Description
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">BaseSQL</a>	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">Connection</a>	Used to specify a connection object to use to connect to a data store.
<a href="#">Debug</a>	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">DetailFields</a>	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#">Disconnected</a>	Used to keep dataset opened after connection is closed.
<a href="#">Encryption</a>	Used to specify the options of the data encryption in a dataset.
<a href="#">FetchRows</a>	Used to define the number of rows to be transferred across the network at the same time.
<a href="#">FilterSQL</a>	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#">FinalSQL</a>	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.

<a href="#">GetBlob</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">IsQuery</a>	Used to check whether SQL statement returns rows.
<a href="#">KeyFields</a>	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">MacroCount</a>	Used to get the number of macros associated with the Macros property.
<a href="#">Macros</a>	Makes it possible to change SQL queries easily.
<a href="#">MasterFields</a>	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#">MasterSource</a>	Used to specify the data source component which binds current dataset to the master one.



<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.
<a href="#">Options</a>	Used to specify the behaviour of TCustomDADataset object.
<a href="#">ParamCheck</a>	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#">ParamCount</a>	Used to indicate how many parameters are there in the Params property.
<a href="#">Params</a>	Used to view and set parameter names, values, and data types dynamically.
<a href="#">Prepare</a> (inherited from <a href="#">TMemDataSet</a> )	Allocates resources and creates field components for a dataset.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">ReadOnly</a>	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#">RefreshOptions</a>	Used to indicate when the editing record is refreshed.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">RowsAffected</a>	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

[SQL](#)

Used to provide a SQL statement that a query component executes when its Open method is called.

[SQLDelete](#)

Used to specify a SQL statement that will be used when applying a deletion to a record.

[SQLInsert](#)

Used to specify the SQL statement that will be used when applying an insertion to a dataset.

[SQLLock](#)

Used to specify a SQL statement that will be used to perform a record lock.

[SQLRefresh](#)

Used to specify a SQL statement that will be used to refresh current record by calling the [TCustomDADataset.RefreshRecord](#) procedure.

[SQLUpdate](#)

Used to specify a SQL statement that will be used when applying an update to a dataset.

[UniDirectional](#)

Used if an application does not need bidirectional access to records in the result set.

[UnPrepare](#) (inherited from [TMemDataSet](#))

Frees the resources allocated for a previously prepared query on the server and client sides.

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

Used to indicate the update status for the current record when cached updates are enabled.

[UpdateResult](#) (inherited from [TMemDataSet](#))

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

[UpdatesPending](#) (inherited from [TMemDataSet](#))

Used to check the status of the cached updates buffer.

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Indicates the current update status for the dataset when cached updates are enabled.

### See Also

- [TCustomDADataset Class](#)
  - [TCustomDADataset Class Members](#)
-

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.2.1 BaseSQL Property

Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.

### Class

[TCustomDADataSet](#)

### Syntax

```
property BaseSQL: string;
```

### Remarks

Use the BaseSQL property to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL, only macros are expanded. SQL text with all these changes can be returned by [FinalSQL](#).

### See Also

- [FinalSQL](#)
- [AddWhere](#)
- [SaveSQL](#)
- [SQLSaved](#)
- [RestoreSQL](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.2.2 Connection Property

Used to specify a connection object to use to connect to a data store.

### Class

[TCustomDADataSet](#)

### Syntax

```
property Connection: TCustomDAConnection;
```

### Remarks

Use the Connection property to specify a connection object that will be used to connect to a data store.

Set at design-time by selecting from the list of provided TCustomDAConnection or its descendant class objects.

At runtime, link an instance of a TCustomDAConnection descendant to the Connection property.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.3 Debug Property

Used to display executing statement, all its parameters' values, and the type of parameters.

**Class**

[TCustomDADataset](#)

**Syntax**

```
property Debug: boolean default False;
```

**Remarks**

Set the Debug property to True to display executing statement and all its parameters' values. Also displays the type of parameters.

You should add the IbDacVcl unit to the uses clause of any unit in your project to make the Debug property work.

**Note:** To enable debug window you should explicitly include the IbDacVcl unit to your project.

**See Also**

- [TCustomDASQL.Debug](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.4 DetailFields Property

Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.

**Class**

[TCustomDADataset](#)

**Syntax**

```
property DetailFields: string;
```

**Remarks**

Use the DetailFields property to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship. DetailFields is a string containing one or more field names in the detail table. Separate field names with semicolons.

Use Field Link Designer to set the value in design time.

**See Also**

- [MasterFields](#)
  - [MasterSource](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.5 Disconnected Property

Used to keep dataset opened after connection is closed.

**Class**

[TCustomDADataset](#)

**Syntax**

```
property Disconnected: boolean;
```

**Remarks**

Set the Disconnected property to True to keep dataset opened after connection is closed.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.6 Encryption Property

Used to specify the options of the data encryption in a dataset.

**Class**

[TCustomDADataset](#)

**Syntax**

```
property Encryption: TDAEncryptionOptions;
```

**Remarks**

Set the properties of Encryption to specify the options of the data encryption in a dataset.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.7 FetchRows Property

Used to define the number of rows to be transferred across the network at the same time.

**Class**

[TCustomDADataset](#)

**Syntax**

```
property FetchRows: integer default 25;
```

**Remarks**

The number of rows that will be transferred across the network at the same time. This property can have a great impact on performance. So it is preferable to choose the optimal value of the FetchRows property for each SQL statement and software/hardware configuration experimentally. The default value is 25.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.8 FilterSQL Property

Used to change the WHERE clause of SELECT statement and reopen a query.

**Class**

[TCustomDataSet](#)

**Syntax**

```
property FilterSQL: string;
```

**Remarks**

The FilterSQL property is similar to the Filter property, but it changes the WHERE clause of SELECT statement and reopens query. Syntax is the same to the WHERE clause.

**Example**

```
Query1.FilterSQL := 'Dept >= 20 and DName LIKE 'M%''';
```

**See Also**

- [AddWhere](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.9 FinalSQL Property

Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.

**Class**

[TCustomDataSet](#)

**Syntax**

```
property FinalSQL: string;
```

**Remarks**

Use FinalSQL to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros. This is the exact statement that will be passed on to the database server.

**See Also**

- [FinalSQL](#)
- [AddWhere](#)
- [SaveSQL](#)
- [SQLSaved](#)
- [RestoreSQL](#)

- [BaseSQL](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.2.10 IsQuery Property

Used to check whether SQL statement returns rows.

### Class

[TCustomDADataset](#)

### Syntax

```
property IsQuery: boolean;
```

### Remarks

After the TCustomDADataset component is prepared, the IsQuery property returns True if SQL statement is a SELECT query.

Use the IsQuery property to check whether the SQL statement returns rows or not. IsQuery is a read-only property. Reading IsQuery on unprepared dataset raises an exception.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.2.11 KeyFields Property

Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.

### Class

[TCustomDADataset](#)

### Syntax

```
property KeyFields: string;
```

### Remarks

TCustomDADataset uses the KeyFields property to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database. For this feature KeyFields may hold a list of semicolon-delimited field names. If KeyFields is not defined before opening dataset, TCustomDADataset requests information about primary keys from server sending an additional query.

KeyFields property may hold the name of a field which will be later assigned with an InterBase sequenced values. Beforehand InterBase generator must be created and its name passed to the [TCustomIBCDataset.KeyGenerator](#) property. Sequences are generated when either Insert or Post method is called. Which of these two methods is used to modify the database is determined by the [TCustomIBCDataset.GeneratorMode](#) property.

**Note:** Though keys may be created across a number of table fields, sequence is generated only for the first field found in the KeyFields property.

### See Also

- [SQLDelete](#)
  - [SQLInsert](#)
  - [SQLRefresh](#)
  - [SQLUpdate](#)
  - [TCustomIBCDDataSet.KeyGenerator](#)
  - [TCustomIBCDDataSet.GeneratorMode](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.2.12 MacroCount Property

Used to get the number of macros associated with the Macros property.

### Class

[TCustomDADataset](#)

### Syntax

```
property MacroCount: word;
```

### Remarks

Use the MacroCount property to get the number of macros associated with the Macros property.

### See Also

- [Macros](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.2.13 Macros Property

Makes it possible to change SQL queries easily.

### Class

[TCustomDADataset](#)

### Syntax

```
property Macros: TMacro stored False;
```

### Remarks

With the help of macros you can easily change SQL query text at design- or runtime. Macros extend abilities of parameters and allow to change conditions in a WHERE clause or sort order in an ORDER BY clause. You just insert &MacroName in the SQL query text and change value of macro in the Macro property editor at design time or call the MacroByName function at run time. At the time of opening the query macro is replaced by its value.

### Example



```
IBCQuery.SQL:= 'SELECT * FROM Dept ORDER BY &Order';  
IBCQuery.MacroByName('Order').Value:= 'DeptNo';  
IBCQuery.Open;
```

## See Also

- [TMacro](#)
- [MacroByName](#)
- [Params](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.2.14 MasterFields Property

Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.

## Class

[TCustomDADataset](#)

## Syntax

```
property MasterFields: string;
```

## Remarks

Use the MasterFields property after setting the [MasterSource](#) property to specify the names of one or more fields that are used as foreign keys for this dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.

MasterFields is a string containing one or more field names in the master table. Separate field names with semicolons.

Each time the current record in the master table changes, the new values in these fields are used to select corresponding records in this table for display.

Use Field Link Designer to set the values at design time after setting the MasterSource property.

## See Also

- [DetailFields](#)
- [MasterSource](#)
- [Master/Detail Relationships](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.2.15 MasterSource Property

Used to specify the data source component which binds current dataset to the master one.

## Class

[TCustomDADataset](#)

## Syntax

```
property MasterSource: TDataSource;
```

## Remarks

The MasterSource property specifies the data source component which binds current dataset to the master one.

TCustomDADataset uses MasterSource to extract foreign key fields values from the master dataset when building master/detail relationship between two datasets. MasterSource must point to another dataset; it cannot point to this dataset component.

When MasterSource is not **nil** dataset fills parameter values with corresponding field values from the current record of the master dataset.

**Note:** Do not set the DataSource property when building master/detail relationships. Although it points to the same object as the MasterSource property, it may lead to undesirable results.

## See Also

- [MasterFields](#)
  - [DetailFields](#)
  - [Master/Detail Relationships](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.2.16 Options Property

Used to specify the behaviour of TCustomDADataset object.

## Class

[TCustomDADataset](#)

## Syntax

```
property Options: TDatasetOptions;
```

## Remarks

Set the properties of Options to specify the behaviour of a TCustomDADataset object.

Descriptions of all options are in the table below.

Option Name	Description
<a href="#">AutoPrepare</a>	Used to execute automatic <a href="#">Prepare</a> on the query execution.
<a href="#">CacheCalcFields</a>	Used to enable caching of the TField. Calculated and TField.Lookup fields.
<a href="#">CompressBlobMode</a>	Used to store values of the BLOB fields in compressed form.

<a href="#"><u>DefaultValues</u></a>	Used to request default values/expressions from the server and assign them to the DefaultExpression property.
<a href="#"><u>DetailDelay</u></a>	Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.
<a href="#"><u>FieldsOrigin</u></a>	Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.
<a href="#"><u>FlatBuffers</u></a>	Used to control how a dataset treats data of the ftString and ftVarBytes fields.
<a href="#"><u>LocalMasterDetail</u></a>	Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.
<a href="#"><u>LongStrings</u></a>	Used to represent string fields with the length that is greater than 255 as TStringField.
<a href="#"><u>NumberRange</u></a>	Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.
<a href="#"><u>QueryRecCount</u></a>	Used for TCustomDADataset to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.
<a href="#"><u>QuoteNames</u></a>	Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.
<a href="#"><u>RemoveOnRefresh</u></a>	Used for a dataset to locally remove a record that can not be found on the server.
<a href="#"><u>RequiredFields</u></a>	Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.
<a href="#"><u>ReturnParams</u></a>	Used to return the new value of fields to dataset after insert or update.
<a href="#"><u>SetFieldsReadOnly</u></a>	Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.
<a href="#"><u>StrictUpdate</u></a>	Used for TCustomDADataset to raise an exception when the number of updated or deleted records is not equal 1.

[TrimFixedChar](#)

Specifies whether to discard all trailing spaces in the string fields of a dataset.

[UpdateAllFields](#)

Used to include all dataset fields in the generated UPDATE and INSERT statements.

[UpdateBatchSize](#)

Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

## See Also

- [Master/Detail Relationships](#)
  - [TMemDataSet.CachedUpdates](#)
- 

©

1997-2013 Devart. All Rights Reserved.

17.10.1.5.2.17 ParamCheck Property

Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.

## Class

[TCustomDADataset](#)

## Syntax

```
property ParamCheck: boolean default True;
```

## Remarks

Use the ParamCheck property to specify whether parameters for the Params property are generated automatically after the SQL property was changed.

Set ParamCheck to True to let dataset automatically generate the Params property for the dataset based on a SQL statement.

Setting ParamCheck to False can be used if the dataset component passes to a server the DDL statements that contain, for example, declarations of stored procedures which themselves will accept parameterized values. The default value is True.

## See Also

- [Params](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.18 ParamCount Property

Used to indicate how many parameters are there in the Params property.

**Class**

[TCustomDADataset](#)

**Syntax**

```
property ParamCount: word;
```

**Remarks**

Use the ParamCount property to determine how many parameters are there in the Params property.

**See Also**

- [Params](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.19 Params Property

Used to view and set parameter names, values, and data types dynamically.

**Class**

[TCustomDADataset](#)

**Syntax**

```
property Params: TDAParams stored False;
```

**Remarks**

Contains the parameters for a query's SQL statement.

Access Params at runtime to view and set parameter names, values, and data types dynamically (at design time use the Parameters editor to set the parameter information). Params is a zero-based array of parameter records. Index specifies the array element to access.

An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName.

**See Also**

- [ParamByName](#)
- [Macros](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.20 ReadOnly Property

Used to prevent users from updating, inserting, or deleting data in the dataset.

**Class**

[TCustomDADataset](#)

**Syntax**

```
property ReadOnly: boolean default False;
```

**Remarks**

Use the ReadOnly property to prevent users from updating, inserting, or deleting data in the dataset. By default, ReadOnly is False, meaning that users can potentially alter data stored in the dataset.

To guarantee that users cannot modify or add data to a dataset, set ReadOnly to True.

When ReadOnly is True, the dataset's CanModify property is False.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.21 RefreshOptions Property

Used to indicate when the editing record is refreshed.

**Class**

[TCustomDADataset](#)

**Syntax**

```
property RefreshOptions: TRefreshOptions default [];
```

**Remarks**

Use the RefreshOptions property to determine when the editing record is refreshed. Refresh is performed by the [RefreshRecord](#) method.

It queries the current record and replaces one in the dataset. Refresh record is useful when the table has triggers or the table fields have default values. Use roBeforeEdit to get actual data before editing.

The default value is [ ].

**See Also**

- [RefreshRecord](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.22 RowsAffected Property

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

**Class**

[TCustomDADataset](#)

## Syntax

**property** RowsAffected: integer;

## Remarks

Check RowsAffected to determine how many rows were inserted, updated, or deleted during the last query operation. If RowsAffected is -1, the query has not inserted, updated, or deleted any rows.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.2.23 SQL Property

Used to provide a SQL statement that a query component executes when its Open method is called.

## Class

[TCustomDADataset](#)

## Syntax

**property** SQL: \_TStrings;

## Remarks

Use the SQL property to provide a SQL statement that a query component executes when its Open method is called. At the design time the SQL property can be edited by invoking the String List editor in Object Inspector.

When SQL is changed, TCustomDADataset calls Close and UnPrepare.

## See Also

- [SQLInsert](#)
- [SQLUpdate](#)
- [SQLDelete](#)
- [SQLRefresh](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.2.24 SQLDelete Property

Used to specify a SQL statement that will be used when applying a deletion to a record.

## Class

[TCustomDADataset](#)

## Syntax

**property** SQLDelete: \_TStrings;

## Remarks

Use the SQLDelete property to specify the SQL statement that will be used when

applying a deletion to a record. Statements can be parameterized queries.  
To create a SQLDelete statement at design-time, use the query statements editor.

### Example

```
DELETE FROM Orders
WHERE
    OrderID = :Old_OrderID
```

### See Also

- [SQL](#)
  - [SQLInsert](#)
  - [SQLUpdate](#)
  - [SQLRefresh](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.2.25 SQLInsert Property

Used to specify the SQL statement that will be used when applying an insertion to a dataset.

### Class

[TCustomDADataSet](#)

### Syntax

```
property SQLInsert: _TStrings;
```

### Remarks

Use the SQLInsert property to specify the SQL statement that will be used when applying an insertion to a dataset. Statements can be parameterized queries. Names of the parameters should be the same as field names. Parameters prefixed with OLD allow using current values of fields prior to the actual operation. Use ReturnParam to return OUT parameters back to dataset.  
To create a SQLInsert statement at design-time, use the query statements editor.

### See Also

- [SQL](#)
  - [SQLUpdate](#)
  - [SQLDelete](#)
  - [SQLRefresh](#)
- 

© 1997-2013 Devart. All Rights Reserved.



## 17.10.1.5.2.26 SQLLock Property

Used to specify a SQL statement that will be used to perform a record lock.

**Class**

[TCustomDADataset](#)

**Syntax**

```
property SQLLock: _TStrings;
```

**Remarks**

Use the SQLLock property to specify a SQL statement that will be used to perform a record lock. Statements can be parameteri ed queries. Names of the parameters should be the same as field names. The parameters prefixed with OLD allow to use current values of fields prior to the actual operation.

To create a SQLLock statement at design-time, the use query statement editor.

**See Also**

- [SQL](#)
- [SQLInsert](#)
- [SQLUpdate](#)
- [SQLDelete](#)
- [SQLRefresh](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.2.27 SQLRefresh Property

Used to specify a SQL statement that will be used to refresh current record by calling the [RefreshRecord](#) procedure.

**Class**

[TCustomDADataset](#)

**Syntax**

```
property SQLRefresh: _TStrings;
```

**Remarks**

Use the SQLRefresh property to specify a SQL statement that will be used to refresh current record by calling the [RefreshRecord](#) procedure.

Different behavior is observed when the SQLRefresh property is assigned with a single WHERE clause that holds frequently altered search condition. In this case the WHERE clause from SQLRefresh is combined with the same clause of the SELECT statement in a SQL property and this final query is then sent to the database server.

To create a SQLRefresh statement at design-time, use the query statements editor.

**Example**

```
SELECT Shipname FROM Orders
WHERE
    OrderID = :OrderID
```

## See Also

- [RefreshRecord](#)
  - [SQL](#)
  - [SQLInsert](#)
  - [SQLUpdate](#)
  - [SQLDelete](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.2.28 SQLUpdate Property

Used to specify a SQL statement that will be used when applying an update to a dataset.

## Class

[TCustomDADataset](#)

## Syntax

```
property SQLUpdate: _TStrings;
```

## Remarks

Use the SQLUpdate property to specify a SQL statement that will be used when applying an update to a dataset. Statements can be parameterized queries. Names of the parameters should be the same as field names. The parameters prefixed with OLD allow to use current values of fields prior to the actual operation.

Use ReturnParam to return OUT parameters back to the dataset.

To create a SQLUpdate statement at design-time, use the query statement editor.

## Example

```
UPDATE Orders
set
    ShipName = :ShipName
WHERE
    OrderID = :Old_OrderID
```

## See Also

- [SQL](#)
  - [SQLInsert](#)
  - [SQLDelete](#)
  - [SQLRefresh](#)
-

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.2.29 UniDirectional Property

Used if an application does not need bidirectional access to records in the result set.

### Class

[TCustomDADataset](#)

### Syntax

```
property UniDirectional: boolean default False;
```

### Remarks

Traditionally SQL cursors are unidirectional. They can travel only forward through a dataset. TCustomDADataset, however, permits bidirectional travelling by caching records. If an application does not need bidirectional access to the records in the result set, set UniDirectional to True. When UniDirectional is True, an application requires less memory and performance is improved. However, UniDirectional datasets cannot be modified.

In FetchAll=False mode data is fetched on demand. When UniDirectional is set to True, data is fetched on demand as well, but obtained rows are not cached except for the current row. So, FetchAll=False mode is a component of UniDirectional=True mode, and setting UniDirectional to True requires FetchAll to be set to False. Pay attention to the restrictions of P:Devart.IbDac.TCustomIBCDataset.FetchAll = False mode.

The default value of UniDirectional is False, enabling forward and backward navigation.

### See Also

- P:Devart.IbDac.TCustomIBCDataset.FetchAll

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.3 Methods

Methods of the **TCustomDADataset** class.

For a complete list of the **TCustomDADataset** class members, see the [TCustomDADataset Members](#) topic.

### Public

Name	Description
<a href="#">AddWhere</a>	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.

<a href="#">BreakExec</a>	Breaks execution of the SQL statement on the server.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">CreateBlobStream</a>	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">DeleteWhere</a>	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#">Execute</a>	Executes a SQL statement on the server.
<a href="#">Executing</a>	Indicates whether SQL statement is still being executed.
<a href="#">Fetched</a>	Used to learn whether TCustomDADataset has already fetched all rows.
<a href="#">Fetching</a>	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#">FetchingAll</a>	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#">FindKey</a>	Searches for a record which contains specified field values.
<a href="#">FindMacro</a>	Indicates whether a specified macro exists in a dataset.
<a href="#">FindNearest</a>	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.

<a href="#">FindParam</a>	Determines if a parameter with the specified name exists in a dataset.
<a href="#">GetBlob</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">GetDataType</a>	Returns internal field types defined in the MemData and accompanying modules.
<a href="#">GetFieldObject</a>	Returns a multireference shared object from field.
<a href="#">GetFieldPrecision</a>	Retrieves the precision of a number field.
<a href="#">GetFieldScale</a>	Retrieves the scale of a number field.
<a href="#">GetOrderBy</a>	Retrieves an ORDER BY clause from a SQL statement.
<a href="#">GotoCurrent</a>	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">Lock</a>	Locks the current record.
<a href="#">MacroByName</a>	Finds a Macro with the name passed in Name.

<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.
<a href="#">ParamByName</a>	Sets or uses parameter information for a specific parameter based on its name.
<a href="#">Prepare</a>	Allocates, opens, and parses cursor for a query.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">RefreshRecord</a>	Actuali es field values for the current record.
<a href="#">RestoreSQL</a>	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a>	Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveSQL</a>	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a>	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQLSaved</a>	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">UnLock</a>	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

Used to indicate the update status for the current record when cached updates are enabled.

[UpdateResult](#) (inherited from [TMemDataSet](#))

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

[UpdatesPending](#) (inherited from [TMemDataSet](#))

Used to check the status of the cached updates buffer.

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Indicates the current update status for the dataset when cached updates are enabled.

### See Also

- [TCustomDADataSet Class](#)
- [TCustomDADataSet Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.3.1 AddWhere Method

Adds condition to the WHERE clause of SELECT statement in the SQL property.

### Class

[TCustomDADataSet](#)

### Syntax

```
procedure AddWhere(Condition: string);
```

#### Parameters

*Condition*

Holds the condition that will be added to the WHERE clause.

### Remarks

Call the AddWhere method to add a condition to the WHERE clause of SELECT statement in the SQL property.

If SELECT has no WHERE clause, AddWhere creates it.

**Note:** The AddWhere method is implicitly called by [RefreshRecord](#). The AddWhere method works for the SELECT statements only.

### See Also

- [DeleteWhere](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.3.2 BreakExec Method

Breaks execution of the SQL statement on the server.

**Class**

[TCustomDataSet](#)

**Syntax**

```
procedure BreakExec; virtual;
```

**Remarks**

Call the BreakExec method to break execution of the SQL statement on the server. It makes sense to call BreakExec only from another thread.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.3.3 CreateBlobStream Method

Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.

**Class**

[TCustomDataSet](#)

**Syntax**

```
function CreateBlobStream(Field: TField; Mode: TBlobStreamMode):  
TStream; override;
```

**Parameters***Field*

Holds the BLOB field for reading data from or writing data to from a stream.

*Mode*

Holds the stream mode, for which the stream will be used.

**Return Value**

The BLOB Stream.

**Remarks**

Call the CreateBlobStream method to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter. It must be a TBlobField component. You can specify whether the stream will be used for reading, writing, or updating the contents of the field with the Mode parameter.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.3.4 DeleteWhere Method

Removes WHERE clause from the SQL property and assigns the BaseSQL property.

**Class**

[TCustomDataSet](#)



## Syntax

```
procedure DeleteWhere;
```

## Remarks

Call the DeleteWhere method to remove WHERE clause from the the SQL property and assign BaseSQL.

## See Also

- [AddWhere](#)
- [BaseSQL](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.3.5 Execute Method

Executes a SQL statement on the server.

## Class

[TCustomDADataset](#)

## Syntax

```
procedure Execute; virtual;
```

## Remarks

Call the Execute method to execute a SQL statement on the server. If SQL statement is a query, Execute calls the Open method.

Execute implicitly prepares SQL statement by calling the [Prepare](#) method if the [Options](#) option is set to True and the statement has not been prepared yet. To speed up the performance in case of multiple Execute calls, an application should call Prepare before calling the Execute method for the first time.

## See Also

- [AfterExecute](#)
- [Executing](#)
- [Prepare](#)
- [TIBCStoredProc.PrepareSQL](#)
- [TIBCStoredProc.Prepare](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.3.6 Executing Method

Indicates whether SQL statement is still being executed.

## Class

[TCustomDADataset](#)

**Syntax**

```
function Executing: boolean;
```

**Return Value**

True, if SQL statement is still being executed.

**Remarks**

Check Executing to learn whether TCustomDADataset is still executing SQL statement. Use the Executing method if NonBlocking is True.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.3.7 Fetched Method

Used to learn whether TCustomDADataset has already fetched all rows.

**Class**

[TCustomDADataset](#)

**Syntax**

```
function Fetched: boolean; virtual;
```

**Return Value**

True, if all rows are fetched.

**Remarks**

Check Fetched to learn whether TCustomDADataset has already fetched all rows.

**See Also**

- [Fetching](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.3.8 Fetching Method

Used to learn whether TCustomDADataset is still fetching rows.

**Class**

[TCustomDADataset](#)

**Syntax**

```
function Fetching: boolean;
```

**Return Value**

True, if TCustomDADataset is still fetching rows.

**Remarks**

Check Fetching to learn whether TCustomDADataset is still fetching rows. Use the Fetching method if NonBlocking is True.

## See Also

- [Executing](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.3.9 FetchingAll Method

Used to learn whether TCustomDADataset is fetching all rows to the end.

## Class

[TCustomDADataset](#)

## Syntax

```
function FetchingAll: boolean;
```

### Return Value

True, if TCustomDADataset is fetching all rows to the end.

## Remarks

Check FetchingAll to learn whether TCustomDADataset is fetching all rows to the end.

## See Also

- [Executing](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.3.10 FindKey Method

Searches for a record which contains specified field values.

## Class

[TCustomDADataset](#)

## Syntax

```
function FindKey(const KeyValues: array of System.TVarRec):  
Boolean;
```

### Parameters

*KeyValues*

Holds a key.

## Remarks

Call the FindKey method to search for a specific record in a dataset. KeyValues holds a comma-delimited array of field values, that is called a key. This function is provided for BDE compatibility only. It is recommended to use functions [TMemDataSet.Locate](#) and [TMemDataSet.LocateEx](#) for the record search.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.3.11 FindMacro Method

Indicates whether a specified macro exists in a dataset.

### Class

[TCustomDADataset](#)

### Syntax

```
function FindMacro(const Value: string): TMacro;
```

#### Parameters

*Value*

Holds the name of the macro to search for.

#### Return Value

a TMacro object, if a macro with matching name was found, otherwise returns nil.

### Remarks

Call the FindMacro method to determine if a specified macro exists. If FindMacro finds a macro with a matching name, it returns a TMacro object for the specified Name. Otherwise it returns nil.

### See Also

- [TMacro](#)
  - [Macros](#)
  - [MacroByName](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.3.12 FindNearest Method

Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.

### Class

[TCustomDADataset](#)

### Syntax

```
procedure FindNearest(const KeyValues: array of System.TVarRec);
```

#### Parameters

*KeyValues*

Holds the values of the record key fields to which the cursor should be moved.

### Remarks

Call the FindNearest method to move the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the

KeyValues parameter. If there are no records that match or exceed the specified criteria, the cursor will not move.  
 This function is provided for BDE compatibility only. It is recommended to use functions [TMemDataSet.Locate](#) and [TMemDataSet.LocateEx](#) for the record search.

## See Also

- [TMemDataSet.Locate](#)
- [TMemDataSet.LocateEx](#)
- [FindKey](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.3.13 FindParam Method

Determines if a parameter with the specified name exists in a dataset.

## Class

[TCustomDADataSet](#)

## Syntax

```
function FindParam(const Value: string): TDAParam;
```

## Parameters

### Value

Holds the name of the param for which to search.

## Return Value

the TDAParam object for the specified Name. Otherwise it returns nil.

## Remarks

Call the FindParam method to determine if a specified param component exists in a dataset. Name is the name of the param for which to search. If FindParam finds a param with a matching name, it returns a TDAParam object for the specified Name. Otherwise it returns nil.

## See Also

- [Params](#)
- [ParamByName](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.3.14 GetDataType Method

Returns internal field types defined in the MemData and accompanying modules.

## Class

[TCustomDADataSet](#)

**Syntax**

```
function GetDataType(const FieldName: string): integer; virtual;
```

**Parameters**

*FieldName*

Holds the name of the field.

**Return Value**

internal field types defined in MemData and accompanying modules.

**Remarks**

Call the GetDataType method to return internal field types defined in the MemData and accompanying modules. Internal field data types extend the TFieldType type of VCL by specific database server data types. For example, ftString, ftFile, ftObject.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.3.15 GetFieldObject Method

Returns a multireference shared object from field.

**Class**

[TCustomDADataset](#)

**Syntax**

```
function GetFieldObject(Field: TField): TSharedObject; overload;  
function GetFieldObject(FieldDesc: TFieldDesc): TSharedObject;  
overload;  
function GetFieldObject(const FieldName: string):  
TSharedObject; overload;
```

**Parameters**

*FieldName*

Holds the field name.

**Return Value**

multireference shared object.

**Remarks**

Call the GetFieldObject method to return a multireference shared object from field. If field does not hold one of the TSharedObject descendants, GetFieldObject raises an exception.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.3.16 GetFieldPrecision Method

Retrieves the precision of a number field.

**Class**

[TCustomDADataset](#)

## Syntax

```
function GetFieldPrecision(const FieldName: string): integer;
```

### Parameters

*FieldName*

Holds the existing field name.

### Return Value

precision of number field.

## Remarks

Call the GetFieldPrecision method to retrieve the precision of a number field. FieldName is the name of an existing field.

## See Also

- [GetFieldScale](#)

---

© 1997-2013 Devart. All Rights Reserved.

17.10.1.5.3.17 GetFieldScale Method

Retrieves the scale of a number field.

## Class

[TCustomDADataset](#)

## Syntax

```
function GetFieldScale(const FieldName: string): integer;
```

### Parameters

*FieldName*

Holds the existing field name.

### Return Value

the scale of the number field.

## Remarks

Call the GetFieldScale method to retrieve the scale of a number field. FieldName is the name of an existing field.

## See Also

- [GetFieldPrecision](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.3.18 GetOrderBy Method

Retrieves an ORDER BY clause from a SQL statement.

**Class**

[TCustomDADataset](#)

**Syntax**

```
function GetOrderBy: string;
```

**Return Value**

an ORDER BY clause from the SQL statement.

**Remarks**

Call the GetOrderBy method to retrieve an ORDER BY clause from a SQL statement.

**Note:** GetOrderBy and SetOrderBy methods serve to process only quite simple queries and don't support, for example, subqueries.

**See Also**

- [SetOrderBy](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.3.19 GotoCurrent Method

Sets the current record in this dataset similar to the current record in another dataset.

**Class**

[TCustomDADataset](#)

**Syntax**

```
procedure GotoCurrent (DataSet: TCustomDADataset);
```

**Parameters**

*DataSet*

Holds the TCustomDADataset descendant to synchronize the record position with.

**Remarks**

Call the GotoCurrent method to set the current record in this dataset similar to the current record in another dataset. The key fields in both these DataSets must be coincident.

**See Also**

- [TMemDataSet.Locate](#)
  - [TMemDataSet.LocateEx](#)
-



© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.3.20 Lock Method

Locks the current record.

### Class

[TCustomDADataset](#)

### Syntax

```
procedure Lock; virtual;
```

### Remarks

Call the Lock method to lock the current record by executing the statement that is defined in the SQLLock property.

The Lock method sets the savepoint with the name LOCK + <component name>.

### See Also

- [Unlock](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.3.21 MacroByName Method

Finds a Macro with the name passed in Name.

### Class

[TCustomDADataset](#)

### Syntax

```
function MacroByName(const Value: string): TMacro;
```

#### Parameters

##### Value

Holds the name of the Macro to search for.

#### Return Value

the Macro, if a match was found.

### Remarks

Call the MacroByName method to find a Macro with the name passed in Name. If a match was found, MacroByName returns the Macro. Otherwise, an exception is raised. Use this method rather than a direct reference to the Items property to avoid depending on the order of the entries.

To locate a parameter by name without raising an exception if the parameter is not found, use the FindMacro method.

To assign the value of macro use the [TMacro.Value](#) property.

## Example

```
IBCQuery.SQL:= 'SELECT * FROM Scott.Dept ORDER BY &Order';  
IBCQuery.MacroByName('Order').Value:= 'DeptNo';  
IBCQuery.Open;
```

## See Also

- [TMacro](#)
  - [Macros](#)
  - [FindMacro](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.5.3.22 ParamByName Method

Sets or uses parameter information for a specific parameter based on its name.

## Class

[TCustomDADataset](#)

## Syntax

```
function ParamByName(const Value: string): TDAParam;
```

### Parameters

#### Value

Holds the name of the parameter for which to retrieve information.

### Return Value

a TDAParam object.

## Remarks

Call the ParamByName method to set or use parameter information for a specific parameter based on its name. Name is the name of the parameter for which to retrieve information. ParamByName is used to set a parameter's value at runtime and returns a [TDAParam](#) object.

## Example

The following statement retrieves the current value of a parameter called "Contact" into an edit box:

```
Edit1.Text := Query1.ParamsByName('Contact').AsString;
```

## See Also

- [Params](#)
- [FindParam](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.3.23 Prepare Method

Allocates, opens, and parses cursor for a query.

### Class

[TCustomDADataset](#)

### Syntax

```
procedure Prepare; override;
```

### Remarks

Call the Prepare method to allocate, open, and parse cursor for a query. Calling Prepare before executing a query improves application performance.

TCustomDADataset automatically prepares a query if it is executed without being prepared first. After execution, TCustomDADataset unprepares the query. When a query is executed a number of times, an application should always explicitly prepare the query to avoid multiple and unnecessary prepares and unprepares. The UnPrepare method unprepares a query.

**Note:** When you change the text of a query at runtime, the query is automatically closed and unprepared.

### See Also

- [TMemDataSet.Prepared](#)
- [TMemDataSet.UnPrepare](#)
- [Options](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.3.24 RefreshRecord Method

Actuali es field values for the current record.

### Class

[TCustomDADataset](#)

### Syntax

```
procedure RefreshRecord;
```

### Remarks

Call the RefreshRecord method to actuali e field values for the current record.

RefreshRecord performs query to database and refetches new field values from the returned cursor.

### See Also

- [RefreshOptions](#)
  - [SQLRefresh](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.3.25 RestoreSQL Method

Restores the SQL property modified by AddWhere and SetOrderBy.

#### Class

[TCustomDataSet](#)

#### Syntax

```
procedure RestoreSQL;
```

#### Remarks

Call the RestoreSQL method to restore the SQL property modified by AddWhere and SetOrderBy.

#### See Also

- [AddWhere](#)
  - [SetOrderBy](#)
  - [SaveSQL](#)
  - [SQLSaved](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.3.26 Resync Method

Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.

#### Class

[TCustomDataSet](#)

#### Syntax

```
procedure Resync (Mode: TResyncMode); override;
```

#### Parameters

*Mode*

Holds optional processing that Resync should handle.

#### Remarks

Resync is used to resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.3.27 SaveSQL Method

Saves the SQL property value to BaseSQL.

**Class**

[TCustomDADataset](#)

**Syntax**

```
procedure SaveSQL;
```

**Remarks**

Call the SaveSQL method to save the SQL property value to the BaseSQL property.

**See Also**

- [SQLSaved](#)
- [RestoreSQL](#)
- [BaseSQL](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.5.3.28 SetOrderBy Method

Builds an ORDER BY clause of a SELECT statement.

**Class**

[TCustomDADataset](#)

**Syntax**

```
procedure SetOrderBy(Fields: string);
```

**Parameters***Fields*

Holds the names of the fields which will be added to the ORDER BY clause.

**Remarks**

Call the SetOrderBy method to build an ORDER BY clause of a SELECT statement. The fields are identified by the comma-delimited field names.

**Note:** The GetOrderBy and SetOrderBy methods serve to process only quite simple queries and don't support, for example, subqueries.

**Example**

```
Query1.SetOrderBy('DeptNo;DName');
```

**See Also**

- [GetOrderBy](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.3.29 SQLSaved Method

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

#### Class

[TCustomDADataset](#)

#### Syntax

```
function SQLSaved: boolean;
```

#### Return Value

True, if the SQL property value was saved to the BaseSQL property.

#### Remarks

Call the SQLSaved method to know whether the [SQL](#) property value was saved to the [BaseSQL](#) property.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.3.30 UnLock Method

Releases a record lock.

#### Class

[TCustomDADataset](#)

#### Syntax

```
procedure UnLock;
```

#### Remarks

Call the Unlock method to release the record lock made by the [Lock](#) method before. Unlock is performed by rolling back to the savepoint set by the Lock method.

#### See Also

- [Lock](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.4 Events

Events of the **TCustomDADataset** class.  
For a complete list of the **TCustomDADataset** class members, see the [TCustomDADataset Members](#) topic.

#### Public

Name	Description
------	-------------

---

<a href="#">AfterExecute</a>	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a>	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a>	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">BeforeFetch</a>	Occurs before dataset is going to fetch block of records from the server.
<a href="#">BeforeUpdateExecute</a>	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">GetBlob</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.

<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.
<a href="#">Prepare</a> (inherited from <a href="#">TMemDataSet</a> )	Allocates resources and creates field components for a dataset.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the <a href="#">ApplyUpdates</a> method while cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.

**See Also**

- [TCustomDADataset Class](#)
- [TCustomDADataset Class Members](#)



---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.4.1 AfterExecute Event

Occurs after a component has executed a query to database.

##### **Class**

[TCustomDADataset](#)

##### **Syntax**

**property** AfterExecute: [TAfterExecuteEvent](#);

##### **Remarks**

Occurs after a component has executed a query to database.

##### **See Also**

- [Execute](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.4.2 AfterFetch Event

Occurs after dataset finishes fetching data from server.

##### **Class**

[TCustomDADataset](#)

##### **Syntax**

**property** AfterFetch: [TAfterFetchEvent](#);

##### **Remarks**

The AfterFetch event occurs after dataset finishes fetching data from server.

##### **See Also**

- [BeforeFetch](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.4.3 AfterUpdateExecute Event

Occurs after executing insert, delete, update, lock and refresh operations.

##### **Class**

[TCustomDADataset](#)

##### **Syntax**

**property** AfterUpdateExecute: [TUpdateExecuteEvent](#);

### Remarks

Occurs after executing insert, delete, update, lock, and refresh operations. You can use AfterUpdateExecute to set the parameters of corresponding statements.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.4.4 BeforeFetch Event

Occurs before dataset is going to fetch block of records from the server.

### Class

[TCustomDADataset](#)

### Syntax

**property** BeforeFetch: [TBeforeFetchEvent](#);

### Remarks

The BeforeFetch event occurs every time before dataset is going to fetch a block of records from the server. Set Cancel to True to abort current fetch operation. To get an example on how to interrupt fetch operation refer to the Loader demo.

### See Also

- [AfterFetch](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.5.4.5 BeforeUpdateExecute Event

Occurs before executing insert, delete, update, lock, and refresh operations.

### Class

[TCustomDADataset](#)

### Syntax

**property** BeforeUpdateExecute: [TUpdateExecuteEvent](#);

### Remarks

Occurs before executing insert, delete, update, lock, and refresh operations. You can use BeforeUpdateExecute to set the parameters of corresponding statements.

### See Also

- [AfterUpdateExecute](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.6 TCustomDASQL Class

A base class for components executing SQL statements that do not return result sets.

For a list of all members of this type, see [TCustomDASQL](#) members.

#### Unit

[DBAccess](#)

#### Syntax

```
TCustomDASQL = class (TComponent) ;
```

#### Remarks

TCustomDASQL is a base class that defines functionality for descendant classes which access database using SQL statements. Applications never use TCustomDASQL objects directly. Instead they use descendants of TCustomDASQL. Use TCustomDASQL when client application must execute SQL statement or call stored procedure on the database server. The SQL statement should not retrieve rows from the database.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.6.1 Members

[TCustomDASQL](#) class overview.

#### Properties

Name	Description
<a href="#">ChangeCursor</a>	Enables or disables changing screen cursor when executing commands in the NonBlocking mode.
<a href="#">Connection</a>	Used to specify a connection object to use to connect to a data store.
<a href="#">Debug</a>	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#">FinalSQL</a>	Used to return a SQL statement with expanded macros.
<a href="#">MacroCount</a>	Used to get the number of macros associated with the Macros property.
<a href="#">Macros</a>	Makes it possible to change SQL queries easily.

[ParamCheck](#)

Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.

[ParamCount](#)

Indicates the number of parameters in the Params property.

[Params](#)

Used to contain parameters for a SQL statement.

[ParamValues](#)

Used to get or set the values of individual field parameters that are identified by name.

[Prepared](#)

Used to indicate whether a query is prepared for execution.

[RowsAffected](#)

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

[SQL](#)

Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.

## Methods

Name	Description
<a href="#"><u>Execute</u></a>	Overloaded. Executes SQL commands.
<a href="#"><u>Executing</u></a>	Checks whether TCustomDASQL still executes a SQL statement.
<a href="#"><u>FindMacro</u></a>	Searches for a macro with the specified name.
<a href="#"><u>FindParam</u></a>	Finds a parameter with the specified name.
<a href="#"><u>MacroByName</u></a>	Finds a Macro with the name passed in Name.
<a href="#"><u>ParamByName</u></a>	Finds a parameter with the specified name.
<a href="#"><u>Prepare</u></a>	Allocates, opens, and parses cursor for a query.
<a href="#"><u>UnPrepare</u></a>	Frees the resources allocated for a previously prepared query on the server and client sides.

[WaitExecuting](#)

Waits until TCustomDASQL executes a SQL statement.

## Events

Name	Description
<a href="#">AfterExecute</a>	Occurs after a SQL statement has been executed.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.6.2 Properties

Properties of the **TCustomDASQL** class.

For a complete list of the **TCustomDASQL** class members, see the [TCustomDASQL Members](#) topic.

## Public

Name	Description
<a href="#">ChangeCursor</a>	Enables or disables changing screen cursor when executing commands in the NonBlocking mode.
<a href="#">Connection</a>	Used to specify a connection object to use to connect to a data store.
<a href="#">Debug</a>	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#">FinalSQL</a>	Used to return a SQL statement with expanded macros.
<a href="#">MacroCount</a>	Used to get the number of macros associated with the Macros property.
<a href="#">Macros</a>	Makes it possible to change SQL queries easily.
<a href="#">ParamCheck</a>	Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.
<a href="#">ParamCount</a>	Indicates the number of parameters in the Params property.
<a href="#">Params</a>	Used to contain parameters for a SQL statement.

[ParamValues](#)

Used to get or set the values of individual field parameters that are identified by name.

[Prepared](#)

Used to indicate whether a query is prepared for execution.

[RowsAffected](#)

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

[SQL](#)

Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.

### See Also

- [TCustomDASQL Class](#)
  - [TCustomDASQL Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.6.2.1 ChangeCursor Property

Enables or disables changing screen cursor when executing commands in the NonBlocking mode.

### Class

[TCustomDASQL](#)

### Syntax

```
property ChangeCursor: boolean;
```

### Remarks

Set the ChangeCursor property to False to prevent the screen cursor from changing to crSQLArrow when executing commands in the NonBlocking mode. The default value is True.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.6.2.2 Connection Property

Used to specify a connection object to use to connect to a data store.

### Class

[TCustomDASQL](#)

### Syntax

```
property Connection: TCustomDAConnection;
```

## Remarks

Use the Connection property to specify a connection object that will be used to connect to a data store.

Set at design-time by selecting from the list of provided TCustomDAConnection or its descendant class objects.

At runtime, link an instance of a TCustomDAConnection descendant to the Connection property.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.6.2.3 Debug Property

Used to display executing statement, all its parameters' values, and the type of parameters.

## Class

[TCustomDASQL](#)

## Syntax

```
property Debug: boolean default False;
```

## Remarks

Set the Debug property to True to display executing statement and all its parameters' values. Also displays the type of parameters.

You should add the IBdacVcl unit to the uses clause of any unit in your project to make the Debug property work.

**Note:** To enable debug window you should explicitly include the IBdacVcl unit to your project.

## See Also

- [TCustomDADataSet.Debug](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.6.2.4 FinalSQL Property

Used to return a SQL statement with expanded macros.

## Class

[TCustomDASQL](#)

## Syntax

```
property FinalSQL: string;
```

## Remarks

Read the FinalSQL property to return a SQL statement with expanded macros. This is the exact statement that will be passed on to the database server.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.6.2.5 MacroCount Property

Used to get the number of macros associated with the Macros property.

### Class

[TCustomDASQL](#)

### Syntax

```
property MacroCount: word;
```

### Remarks

Use the MacroCount property to get the number of macros associated with the Macros property.

### See Also

- [Macros](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.6.2.6 Macros Property

Makes it possible to change SQL queries easily.

### Class

[TCustomDASQL](#)

### Syntax

```
property Macros: TMacros stored False;
```

### Remarks

With the help of macros you can easily change SQL query text at design- or runtime. Macros extend abilities of parameters and allow to change conditions in a WHERE clause or sort order in an ORDER BY clause. You just insert &MacroName in the SQL query text and change value of macro in the Macro property editor at design time or call the MacroByName function at run time. At the time of opening the query macro is replaced by its value.

### See Also

- [TMacro](#)
  - [MacroByName](#)
  - [Params](#)
- 

© 1997-2013 Devart. All Rights Reserved.



## 17.10.1.6.2.7 ParamCheck Property

Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.

**Class**

[TCustomDASQL](#)

**Syntax**

```
property ParamCheck: boolean default True;
```

**Remarks**

Use the ParamCheck property to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed. Set ParamCheck to True to let TCustomDASQL generate the Params property for the dataset based on a SQL statement automatically. Setting ParamCheck to False can be used if the dataset component passes to a server the DDL statements that contain, for example, declarations of the stored procedures that will accept parameterized values themselves. The default value is True.

**See Also**

- [Params](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.6.2.8 ParamCount Property

Indicates the number of parameters in the Params property.

**Class**

[TCustomDASQL](#)

**Syntax**

```
property ParamCount: word;
```

**Remarks**

Use the ParamCount property to determine how many parameters are there in the Params property.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.6.2.9 Params Property

Used to contain parameters for a SQL statement.

**Class**

[TCustomDASQL](#)

**Syntax**

```
property Params: TDAParams stored False;
```

### Remarks

Access the Params property at runtime to view and set parameter names, values, and data types dynamically (at design-time use the Parameters editor to set parameter properties). Params is a zero-based array of parameter records. Index specifies the array element to access. An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName.

### Example

Setting parameters at runtime:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with IBSQL do
    begin
      SQL.Clear;
      SQL.Add('INSERT INTO Temp Table(Id, Name)');
      SQL.Add('VALUES (:id, :Name)');
      ParamByName('Id').AsInteger := 55;
      Params[1].AsString := ' Green';
      Execute;
    end;
end;
```

### See Also

- [TDAParam](#)
- [FindParam](#)
- [Macros](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.6.2.10 ParamValues Property(Indexer)

Used to get or set the values of individual field parameters that are identified by name.

### Class

[TCustomDASQL](#)

### Syntax

```
property ParamValues[ParamName: string]: variant; default;
```

**Parameters**

*ParamName*

Holds parameter names separated by semicolon.

### Remarks

Use the ParamValues property to get or set the values of individual field parameters

that are identified by name.

Setting ParamValues sets the Value property for each parameter listed in the ParamName string. Specify the values as Variants.

Getting ParamValues retrieves an array of variants, each of which represents the value of one of the named parameters.

**Note:** The Params array is generated implicitly if ParamCheck property is set to True. If ParamName includes a name that does not match any of the parameters in Items, an exception is raised.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.6.2.11 Prepared Property

Used to indicate whether a query is prepared for execution.

### Class

[TCustomDASQL](#)

### Syntax

```
property Prepared: boolean;
```

### Remarks

Check the Prepared property to determine if a query is already prepared for execution. True means that the query has already been prepared. As a rule prepared queries are executed faster, but the preparation itself also takes some time. One of the proper cases for using preparation is parametrized queries that are executed several times.

### See Also

- [Prepare](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.6.2.12 RowsAffected Property

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

### Class

[TCustomDASQL](#)

### Syntax

```
property RowsAffected: integer;
```

### Remarks

Check RowsAffected to determine how many rows were inserted, updated, or deleted during the last query operation. If RowsAffected is -1, the query has not inserted, updated, or deleted any rows.

## Example

For correct initiali ing this property you should explicitly prepare SQL as it is shown in the example below.

```
IBCSQL.SQL.Text := 'Update Employee set salary = :sal where emp no = :no';
IBCSQL.Prepare;
IBCSQL.Execute;
AffRows := IBSQL.RowsAffected;
IBCSQL.Unprepare;
```

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.6.2.13 SQL Property

Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.

## Class

[TCustomDASQL](#)

## Syntax

**property** SQL: \_TStrings;

## Remarks

Use the SQL property to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called. At design time the SQL property can be edited by invoking the String List editor in Object Inspector.

## See Also

- [FinalSQL](#)
- [TCustomDASQL.Execute](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.6.3 Methods

Methods of the **TCustomDASQL** class.

For a complete list of the **TCustomDASQL** class members, see the [TCustomDASQL Members](#) topic.

## Public

Name	Description
<a href="#">Execute</a>	Overloaded. Executes SQL commands.
<a href="#">Executing</a>	Checks whether TCustomDASQL still executes a SQL statement.
<a href="#">FindMacro</a>	Searches for a macro with the specified name.

[FindParam](#)

Finds a parameter with the specified name.

[MacroByName](#)

Finds a Macro with the name passed in Name.

[ParamByName](#)

Finds a parameter with the specified name.

[Prepare](#)

Allocates, opens, and parses cursor for a query.

[UnPrepare](#)

Frees the resources allocated for a previously prepared query on the server and client sides.

[WaitExecuting](#)

Waits until TCustomDASQL executes a SQL statement.

### See Also

- [TCustomDASQL Class](#)
- [TCustomDASQL Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.6.3.1 Execute Method

Executes SQL commands.

### Class

[TCustomDASQL](#)

### Overload List

Name	Description
<a href="#">Execute</a>	Executes SQL commands.
<a href="#">Execute(Iter: integer)</a>	Is not used in IBDAC.

© 1997-2013 Devart. All Rights Reserved.

Executes SQL commands.

### Class

[TCustomDASQL](#)

### Syntax

**procedure** Execute; **overload**; **virtual**

### Remarks

Call the Execute method to execute a SQL statement on the server. If the SQL statement has OUT parameters, use the [TCustomDASQL.ParamByName](#) method or the [TCustomDASQL.Params](#) property to get their values. Iters argument specifies the number of times this statement is executed for the DML array operations.

© 1997-2013 Devart. All Rights Reserved.

Is not used in IBDAC.

## Class

[TCustomDASQL](#)

## Syntax

```
procedure Execute(Iter: integer); overload; virtual
```

### Parameters

*Iter*

Is not used in IBDAC.

## Remarks

Is not used in IBDAC.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.6.3.2 Executing Method

Checks whether TCustomDASQL still executes a SQL statement.

## Class

[TCustomDASQL](#)

## Syntax

```
function Executing: boolean;
```

### Return Value

True, if a SQL statement is still being executed by TCustomDASQL.

## Remarks

Check Executing to find out whether TCustomDASQL still executes a SQL statement. Executing method is used for nonblocking execution.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.6.3.3 FindMacro Method

Searches for a macro with the specified name.

## Class

[TCustomDASQL](#)

## Syntax

```
function FindMacro(const Value: string): TMacro;
```

### Parameters

*Value*

Holds the name of a macro to search for.

### Return Value

the TMacro object, if a macro with the specified name has been found. If it has

not, returns nil.

### Remarks

Call the FindMacro method to find a macro with the specified name in a dataset.

### See Also

- [TMacro](#)
- [Macros](#)
- [MacroByName](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.6.3.4 FindParam Method

Finds a parameter with the specified name.

### Class

[TCustomDASQL](#)

### Syntax

```
function FindParam(const Value: string): TDAParm;
```

#### Parameters

*Value*

Holds the parameter name to search for.

#### Return Value

a TDAParm object, if a parameter with the specified name has been found. If it has not, returns nil.

### Remarks

Call the FindParam method to find a parameter with the specified name in a dataset.

### See Also

- [ParamByName](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.6.3.5 MacroByName Method

Finds a Macro with the name passed in Name.

### Class

[TCustomDASQL](#)

### Syntax

```
function MacroByName(const Value: string): TMacro;
```

**Parameters***Value*

Holds the name of the Macro to search for.

**Return Value**

the Macro, if a match was found.

**Remarks**

Call the MacroByName method to find a Macro with the name passed in Name. If a match was found, MacroByName returns the Macro. Otherwise, an exception is raised. Use this method rather than a direct reference to the Items property to avoid depending on the order of the entries.

To locate a parameter by name without raising an exception if the parameter is not found, use the FindMacro method.

To assign the value of macro use the [TMacro.Value](#) property.

**See Also**

- [TMacro](#)
  - [Macros](#)
  - [FindMacro](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.6.3.6 ParamByName Method

Finds a parameter with the specified name.

**Class**

[TCustomDASQL](#)

**Syntax**

```
function ParamByName(const Value: string): TDAParam;
```

**Parameters***Value*

Holds the name of the parameter to search for.

**Return Value**

a TDAParam object, if a match was found. Otherwise, an exception is raised.

**Remarks**

Use the ParamByName method to find a parameter with the specified name. If no parameter with the specified name found, an exception is raised.

**Example**

```
IBCSQL.Execute;
```



```
Edit1.Text := IBCSQL.ParamsByName('Contact').AsString;
```

## See Also

- 

[FindParam](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.6.3.7 Prepare Method

Allocates, opens, and parses cursor for a query.

## Class

[TCustomDASQL](#)

## Syntax

```
procedure Prepare; virtual;
```

## Remarks

Call the Prepare method to allocate, open, and parse cursor for a query. Calling Prepare before executing a query improves application performance.

TCustomDADataset automatically prepares a query if it is executed without being prepared first. After execution, TCustomDADataset unprepares the query. When a query is executed a number of times, an application should always explicitly prepare the query to avoid multiple and unnecessary prepares and unprepares.

The UnPrepare method unprepares a query.

**Note:** When you change the text of a query at runtime, the query is automatically closed and unprepared.

## See Also

- [Prepared](#)
- [UnPrepare](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.6.3.8 UnPrepare Method

Frees the resources allocated for a previously prepared query on the server and client sides.

## Class

[TCustomDASQL](#)

## Syntax

```
procedure UnPrepare; virtual;
```

## Remarks

Call the UnPrepare method to free resources allocated for a previously prepared

query on the server and client sides.

## See Also

- [Prepare](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.6.3.9 WaitExecuting Method

Waits until TCustomDASQL executes a SQL statement.

## Class

[TCustomDASQL](#)

## Syntax

```
function WaitExecuting(TimeOut: integer = 0): boolean;
```

### Parameters

#### *TimeOut*

Holds the time in seconds to wait while TCustomDASQL executes the SQL statement. Zero means infinite time.

### Return Value

True, if the execution of a SQL statement was completed in the preset time.

## Remarks

Call the WaitExecuting method to wait until TCustomDASQL executes a SQL statement. Use the WaitExecuting method for nonblocking execution.

## See Also

- [Executing](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.6.4 Events

Events of the **TCustomDASQL** class.

For a complete list of the **TCustomDASQL** class members, see the [TCustomDASQL Members](#) topic.

## Public

Name	Description
<a href="#">AfterExecute</a>	Occurs after a SQL statement has been executed.

## See Also

- [TCustomDASQL Class](#)
- [TCustomDASQL Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.6.4.1 AfterExecute Event

Occurs after a SQL statement has been executed.

### Class

[TCustomDASQL](#)

### Syntax

```
property AfterExecute: TAfterExecuteEvent;
```

### Remarks

Occurs after a SQL statement has been executed. This event may be used for descendant components which use multithreaded environment.

### See Also

- [TCustomDASQL.Execute](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.7 TCustomDAUpdateSQL Class

A base class for components that provide DML statements for more flexible control over data modifications.

For a list of all members of this type, see [TCustomDAUpdateSQL](#) members.

### Unit

[DBAccess](#)

### Syntax

```
TCustomDAUpdateSQL = class (TComponent);
```

### Remarks

TCustomDAUpdateSQL is a base class for components that provide DML statements for more flexible control over data modifications. Besides providing BDE compatibility, this component allows to associate a separate component for each update command.

### See Also

- P:Devart.IbDac.TCustomIBCDDataSet.UpdateObject

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.7.1 Members

[TCustomDAUpdateSQL](#) class overview.

**Properties**

Name	Description
<a href="#">DataSet</a>	Used to hold a reference to the TCustomDADataSet object that is being updated.
<a href="#">DeleteObject</a>	Provides ability to perform advanced adjustment of the delete operations.
<a href="#">DeleteSQL</a>	Used when deleting a record.
<a href="#">InsertObject</a>	Provides ability to perform advanced adjustment of insert operations.
<a href="#">InsertSQL</a>	Used when inserting a record.
<a href="#">LockObject</a>	Provides ability to perform advanced adjustment of lock operations.
<a href="#">LockSQL</a>	Used to lock the current record.
<a href="#">ModifyObject</a>	Provides ability to perform advanced adjustment of modify operations.
<a href="#">ModifySQL</a>	Used when updating a record.
<a href="#">RefreshObject</a>	Provides ability to perform advanced adjustment of refresh operations.
<a href="#">RefreshSQL</a>	Used to specify an SQL statement that will be used for refreshing the current record by <a href="#">TCustomDADataSet.RefreshRecord</a> procedure.
<a href="#">SQL</a>	Used to return a SQL statement for one of the ModifySQL, InsertSQL, or DeleteSQL properties.

**Methods**

Name	Description
<a href="#">Apply</a>	Sets parameters for a SQL statement and executes it to update a record.
<a href="#">ExecSQL</a>	Executes a SQL statement.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.7.2 Properties

Properties of the **TCustomDAUpdateSQL** class.

For a complete list of the **TCustomDAUpdateSQL** class members, see the [TCustomDAUpdateSQL Members](#) topic.

#### Public

Name	Description
<a href="#">DataSet</a>	Used to hold a reference to the TCustomDADataSet object that is being updated.
<a href="#">SQL</a>	Used to return a SQL statement for one of the ModifySQL, InsertSQL, or DeleteSQL properties.

#### Published

Name	Description
<a href="#">DeleteObject</a>	Provides ability to perform advanced adjustment of the delete operations.
<a href="#">DeleteSQL</a>	Used when deleting a record.
<a href="#">InsertObject</a>	Provides ability to perform advanced adjustment of insert operations.
<a href="#">InsertSQL</a>	Used when inserting a record.
<a href="#">LockObject</a>	Provides ability to perform advanced adjustment of lock operations.
<a href="#">LockSQL</a>	Used to lock the current record.
<a href="#">ModifyObject</a>	Provides ability to perform advanced adjustment of modify operations.
<a href="#">ModifySQL</a>	Used when updating a record.
<a href="#">RefreshObject</a>	Provides ability to perform advanced adjustment of refresh operations.
<a href="#">RefreshSQL</a>	Used to specify an SQL statement that will be used for refreshing the current record by <a href="#">TCustomDADataSet.RefreshRecord</a> procedure.

**See Also**

- [TCustomDAUpdateSQL Class](#)
  - [TCustomDAUpdateSQL Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.7.2.1 DataSet Property

Used to hold a reference to the TCustomDADataset object that is being updated.

**Class**

[TCustomDAUpdateSQL](#)

**Syntax**

```
property DataSet: TCustomDADataset;
```

**Remarks**

The DataSet property holds a reference to the TCustomDADataset object that is being updated. Generally it is not used directly.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.7.2.2 DeleteObject Property

Provides ability to perform advanced adjustment of the delete operations.

**Class**

[TCustomDAUpdateSQL](#)

**Syntax**

```
property DeleteObject: TComponent;
```

**Remarks**

Assign SQL component or a TCustomIBCDataset descendant to this property to perform advanced adjustment of the delete operations. In some cases this can give some additional performance. Use the same principle to set the SQL property of an object as for setting the [DeleteSQL](#) property.

**See Also**

- [DeleteSQL](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.7.2.3 DeleteSQL Property

Used when deleting a record.

**Class**

[TCustomDAUpdateSQL](#)

**Syntax**

```
property DeleteSQL: _TStrings;
```

**Remarks**

Set the DeleteSQL property to a DELETE statement to use when deleting a record. Statements can be parameterized queries with parameter names corresponding to the dataset field names.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.7.2.4 InsertObject Property

Provides ability to perform advanced adjustment of insert operations.

**Class**

[TCustomDAUpdateSQL](#)

**Syntax**

```
property InsertObject: TComponent;
```

**Remarks**

Assign SQL component or TCustomIBCDataset descendant to this property to perform advanced adjustment of insert operations. In some cases this can give some additional performance. Set the SQL property of the object in the same way as used for the [InsertSQL](#) property.

**See Also**

- [InsertSQL](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.7.2.5 InsertSQL Property

Used when inserting a record.

**Class**

[TCustomDAUpdateSQL](#)

**Syntax**

```
property InsertSQL: _TStrings;
```

**Remarks**

Set the InsertSQL property to an INSERT INTO statement to use when inserting a record. Statements can be parameterized queries with parameter names corresponding to the dataset field names.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.7.2.6 LockObject Property

Provides ability to perform advanced adjustment of lock operations.

**Class**

[TCustomDAUpdateSQL](#)

**Syntax**

```
property LockObject: TComponent;
```

**Remarks**

Assign a SQL component or TCustomIBCDataset descendant to this property to perform advanced adjustment of lock operations. In some cases that can give some additional performance. Set the SQL property of an object in the same way as used for the [LockSQL](#) property.

**See Also**

- [LockSQL](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.7.2.7 LockSQL Property

Used to lock the current record.

**Class**

[TCustomDAUpdateSQL](#)

**Syntax**

```
property LockSQL: _TStrings;
```

**Remarks**

Use the LockSQL property to lock the current record. Statements can be parameterized queries with parameter names corresponding to the dataset field names.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.7.2.8 ModifyObject Property

Provides ability to perform advanced adjustment of modify operations.

**Class**

[TCustomDAUpdateSQL](#)

**Syntax**

```
property ModifyObject: TComponent;
```

**Remarks**



Assign a SQL component or TCustomIBCDataset descendant to this property to perform advanced adjustment of modify operations. In some cases this can give some additional performance. Set the SQL property of the object in the same way as used for the [ModifySQL](#) property.

## See Also

- [ModifySQL](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.7.2.9 ModifySQL Property

Used when updating a record.

## Class

[TCustomDAUpdateSQL](#)

## Syntax

```
property ModifySQL: _TStrings;
```

## Remarks

Set ModifySQL to an UPDATE statement to use when updating a record. Statements can be parameterized queries with parameter names corresponding to the dataset field names.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.7.2.10 RefreshObject Property

Provides ability to perform advanced adjustment of refresh operations.

## Class

[TCustomDAUpdateSQL](#)

## Syntax

```
property RefreshObject: TComponent;
```

## Remarks

Assign a SQL component or TCustomIBCDataset descendant to this property to perform advanced adjustment of refresh operations. In some cases that can give some additional performance. Set the SQL property of the object in the same way as used for the [RefreshSQL](#) property.

## See Also

- [RefreshSQL](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.7.2.11 RefreshSQL Property

Used to specify an SQL statement that will be used for refreshing the current record by [TCustomDADataset.RefreshRecord](#) procedure.

**Class**

[TCustomDAUpdateSQL](#)

**Syntax**

```
property RefreshSQL: _TStrings;
```

**Remarks**

Use the RefreshSQL property to specify a SQL statement that will be used for refreshing the current record by the [TCustomDADataset.RefreshRecord](#) procedure. You can assign to SQLRefresh a WHERE clause only. In such a case it is added to SELECT defined by the SQL property by [TCustomDADataset.AddWhere](#). To create a RefreshSQL statement at design time, use the query statements editor.

**See Also**

- [TCustomDADataset.RefreshRecord](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.7.2.12 SQL Property(Indexer)

Used to return a SQL statement for one of the ModifySQL, InsertSQL, or DeleteSQL properties.

**Class**

[TCustomDAUpdateSQL](#)

**Syntax**

```
property SQL[UpdateKind: TUpdateKind]: _TStrings;
```

**Parameters**

*UpdateKind*

Specifies which of update SQL statements to return.

**Remarks**

Returns a SQL statement for one of the ModifySQL, InsertSQL, or DeleteSQL properties, depending on the value of the UpdateKind index.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.7.3 Methods

Methods of the **TCustomDAUpdateSQL** class.  
For a complete list of the **TCustomDAUpdateSQL** class members, see the [TCustomDAUpdateSQL Members](#) topic.

## Public

Name	Description
<a href="#">Apply</a>	Sets parameters for a SQL statement and executes it to update a record.
<a href="#">ExecSQL</a>	Executes a SQL statement.

## See Also

- [TCustomDAUpdateSQL Class](#)
- [TCustomDAUpdateSQL Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.7.3.1 Apply Method

Sets parameters for a SQL statement and executes it to update a record.

## Class

[TCustomDAUpdateSQL](#)

## Syntax

```
procedure Apply(UpdateKind: TUpdateKind); virtual;
```

### Parameters

*UpdateKind*

Specifies which of update SQL statements to execute.

## Remarks

Call the Apply method to set parameters for a SQL statement and execute it to update a record. UpdateKind indicates which SQL statement to bind and execute. Apply is primarily intended for manually executing update statements from an OnUpdateRecord event handler.

**Note:** If a SQL statement does not contain parameters, it is more efficient to call ExecSQL instead of Apply.

## See Also

- [ExecSQL](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.7.3.2 ExecSQL Method

Executes a SQL statement.

## Class

[TCustomDAUpdateSQL](#)

## Syntax

```
procedure ExecSQL(UpdateKind: TUpdateKind);
```

### Parameters

#### *UpdateKind*

Specifies the kind of update statement to be executed.

### Remarks

Call the ExecSQL method to execute a SQL statement, necessary for updating the records belonging to a read-only result set when cached updates is enabled.

UpdateKind specifies the statement to execute.

ExecSQL is primarily intended for manually executing update statements from the OnUpdateRecord event handler.

**Note:** To both bind parameters and execute a statement, call [Apply](#).

### See Also

- [Apply](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.8 TDAConnectionOptions Class

This class allows setting up the behaviour of the TDAConnection class.

For a list of all members of this type, see [TDAConnectionOptions](#) members.

### Unit

[DBAccess](#)

### Syntax

```
TDAConnectionOptions = class(TPersistent);
```

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.8.1 Members

[TDAConnectionOptions](#) class overview.

### Properties

Name	Description
<a href="#">DefaultSortType</a>	Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the <a href="#">TMemDataSet.IndexFieldNames</a> property of a dataset.

[DisconnectedMode](#)

Used to open a connection only when needed for performing a server call and closes after performing the operation.

[KeepDesignConnected](#)

Used to prevent an application from establishing a connection at the time of startup.

[LocalFailover](#)

If True, the [TCustomDAConnection.OnConnectionLost](#) event occurs and a failover operation can be performed after connection breaks.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.8.2 Properties

Properties of the **TDACConnectionOptions** class.

For a complete list of the **TDACConnectionOptions** class members, see the [TDACConnectionOptions Members](#) topic.

**Public**

Name	Description
<a href="#"><u>DefaultSortType</u></a>	Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the <a href="#"><u>TMemDataSet.IndexFieldNames</u></a> property of a dataset.
<a href="#"><u>DisconnectedMode</u></a>	Used to open a connection only when needed for performing a server call and closes after performing the operation.
<a href="#"><u>KeepDesignConnected</u></a>	Used to prevent an application from establishing a connection at the time of startup.
<a href="#"><u>LocalFailover</u></a>	If True, the <a href="#"><u>TCustomDAConnection.OnConnectionLost</u></a> event occurs and a failover operation can be performed after connection breaks.

**See Also**

- [TDAConnectionOptions Class](#)
  - [TDAConnectionOptions Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.8.2.1 DefaultSortType Property

Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the [TMemDataSet.IndexFieldNames](#) property of a dataset.

**Class**

[TDAConnectionOptions](#)

**Syntax**

```
property DefaultSortType: TSortType default stCaseSensitive;
```

**Remarks**

Use the DefaultSortType property to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the [TMemDataSet.IndexFieldNames](#) property of a dataset.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.8.2.2 DisconnectedMode Property

Used to open a connection only when needed for performing a server call and closes after performing the operation.

**Class**

[TDAConnectionOptions](#)

**Syntax**

```
property DisconnectedMode: boolean default False;
```

**Remarks**

If True, connection opens only when needed for performing a server call and closes after performing the operation. Datasets remain opened when connection closes. May be useful to save server resources and operate in unstable or expensive network. Drawback of using disconnect mode is that each connection establishing requires some time for authentication. If connection is often closed and opened it can slow down the application work. See the [Disconnected Mode](#) topic for more information.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.8.2.3 KeepDesignConnected Property

Used to prevent an application from establishing a connection at the time of startup.

## Class

[TDAConnectionOptions](#)

## Syntax

```
property KeepDesignConnected: boolean default True;
```

## Remarks

At the time of startup prevents application from establishing a connection even if the Connected property was set to True at design-time. Set KeepDesignConnected to False to initialize the connected property to False, even if it was True at design-time.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.8.2.4 LocalFailover Property

If True, the [TCustomDAConnection.OnConnectionLost](#) event occurs and a failover operation can be performed after connection breaks.

## Class

[TDAConnectionOptions](#)

## Syntax

```
property LocalFailover: boolean default False;
```

## Remarks

If True, the [TCustomDAConnection.OnConnectionLost](#) event occurs and a failover operation can be performed after connection breaks. Read the [Working in an Unstable Network](#) topic for more information about using failover.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.9 TDADatasetOptions Class

This class allows setting up the behaviour of the TDADataset class.  
For a list of all members of this type, see [TDADatasetOptions](#) members.

## Unit

[DBAccess](#)

## Syntax

```
TDADatasetOptions = class (TPersistent);
```

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.9.1 Members

[TDADatasetOptions](#) class overview.

## Properties

Name	Description
<a href="#">AutoPrepare</a>	Used to execute automatic <a href="#">TCustomDADataset.Prepare</a> on the query execution.
<a href="#">CacheCalcFields</a>	Used to enable caching of the TField.Calculated and TField.Lookup fields.
<a href="#">CompressBlobMode</a>	Used to store values of the BLOB fields in compressed form.
<a href="#">DefaultValues</a>	Used to request default values/expressions from the server and assign them to the DefaultExpression property.
<a href="#">DetailDelay</a>	Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.
<a href="#">FieldsOrigin</a>	Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.
<a href="#">FlatBuffers</a>	Used to control how a dataset treats data of the ftString and ftVarBytes fields.
<a href="#">LocalMasterDetail</a>	Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.
<a href="#">LongStrings</a>	Used to represent string fields with the length that is greater than 255 as TStringField.
<a href="#">NumberRange</a>	Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.



[QueryRecCount](#)

Used for TCustomDADataset to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.

[QuoteNames](#)

Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.

[RemoveOnRefresh](#)

Used for a dataset to locally remove a record that can not be found on the server.

[RequiredFields](#)

Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.

[ReturnParams](#)

Used to return the new value of fields to dataset after insert or update.

[SetFieldsReadOnly](#)

Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.

[StrictUpdate](#)

Used for TCustomDADataset to raise an exception when the number of updated or deleted records is not equal 1.

[TrimFixedChar](#)

Specifies whether to discard all trailing spaces in the string fields of a dataset.

[UpdateAllFields](#)

Used to include all dataset fields in the generated UPDATE and INSERT statements.

[UpdateBatchSize](#)

Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

## 17.10.1.9.2 Properties

Properties of the **TDADatasetOptions** class.

For a complete list of the **TDADatasetOptions** class members, see the [TDADatasetOptions Members](#) topic.

**Public**

Name	Description
<a href="#">AutoPrepare</a>	Used to execute automatic <a href="#">TCustomDADataset.Prepare</a> on the query execution.
<a href="#">CacheCalcFields</a>	Used to enable caching of the TField.Calculated and TField.Lookup fields.
<a href="#">CompressBlobMode</a>	Used to store values of the BLOB fields in compressed form.
<a href="#">DefaultValues</a>	Used to request default values/expressions from the server and assign them to the DefaultExpression property.
<a href="#">DetailDelay</a>	Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.
<a href="#">FieldsOrigin</a>	Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.
<a href="#">FlatBuffers</a>	Used to control how a dataset treats data of the ftString and ftVarBytes fields.
<a href="#">LocalMasterDetail</a>	Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.
<a href="#">LongStrings</a>	Used to represent string fields with the length that is greater than 255 as TStringField.
<a href="#">NumberRange</a>	Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.

[QueryRecCount](#)

Used for TCustomDADataset to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.

[QuoteNames](#)

Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.

[RemoveOnRefresh](#)

Used for a dataset to locally remove a record that can not be found on the server.

[RequiredFields](#)

Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.

[ReturnParams](#)

Used to return the new value of fields to dataset after insert or update.

[SetFieldsReadOnly](#)

Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.

[StrictUpdate](#)

Used for TCustomDADataset to raise an exception when the number of updated or deleted records is not equal 1.

[TrimFixedChar](#)

Specifies whether to discard all trailing spaces in the string fields of a dataset.

[UpdateAllFields](#)

Used to include all dataset fields in the generated UPDATE and INSERT statements.

[UpdateBatchSize](#)

Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

**See Also**

- [TDADatasetOptions Class](#)
- [TDADatasetOptions Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.9.2.1 AutoPrepare Property

Used to execute automatic [TCustomDADataset.Prepare](#) on the query execution.

##### Class

[TDADatasetOptions](#)

##### Syntax

```
property AutoPrepare: boolean default False;
```

##### Remarks

Use the AutoPrepare property to execute automatic [TCustomDADataset.Prepare](#) on the query execution. Makes sense for cases when a query will be executed several times, for example, in Master/Detail relationships.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.9.2.2 CacheCalcFields Property

Used to enable caching of the TField.Calculated and TField.Lookup fields.

##### Class

[TDADatasetOptions](#)

##### Syntax

```
property CacheCalcFields: boolean default False;
```

##### Remarks

Use the CacheCalcFields property to enable caching of the TField.Calculated and TField.Lookup fields. It can be useful for reducing CPU usage for calculated fields. Using caching of calculated and lookup fields increases memory usage on the client side.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.9.2.3 CompressBlobMode Property

Used to store values of the BLOB fields in compressed form.

##### Class

[TDADatasetOptions](#)

##### Syntax

```
property CompressBlobMode: TCompressBlobMode default cbNone;
```

##### Remarks

Use the CompressBlobMode property to store values of the BLOB fields in compressed form. Add the MemData unit to uses list to use this option. Compression rate greatly depends on stored data, for example, usually graphic data

compresses badly unlike text.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.9.2.4 DefaultValue Property

Used to request default values/expressions from the server and assign them to the DefaultExpression property.

##### Class

[TDADatasetOptions](#)

##### Syntax

```
property DefaultValue: boolean default False;
```

##### Remarks

If True, the default values/expressions are requested from the server and assigned to the DefaultExpression property of TField objects replacing already existent values.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.9.2.5 DetailDelay Property

Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.

##### Class

[TDADatasetOptions](#)

##### Syntax

```
property DetailDelay: integer default 0;
```

##### Remarks

Use the DetailDelay property to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset. If DetailDelay is 0 (the default value) then refreshing of detail dataset occurs immediately. The DetailDelay option should be used for detail dataset.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.9.2.6 FieldsOrigin Property

Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.

##### Class

[TDADatasetOptions](#)

##### Syntax

```
property FieldsOrigin: boolean default False;
```

**Remarks**

If True, TCustomDADataset fills the Origin property of the TField objects by appropriate value when opening a dataset.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.9.2.7 FlatBuffers Property

Used to control how a dataset treats data of the ftString and ftVarBytes fields.

**Class**

[TDADatasetOptions](#)

**Syntax**

```
property FlatBuffers: boolean default False;
```

**Remarks**

Use the FlatBuffers property to control how a dataset treats data of the ftString and ftVarBytes fields. When set to True, all data fetched from the server is stored in record pdata without unused tails.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.9.2.8 LocalMasterDetail Property

Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.

**Class**

[TDADatasetOptions](#)

**Syntax**

```
property LocalMasterDetail: boolean default False;
```

**Remarks**

If True, for detail dataset in master-detail relationship TCustomDADataset uses local filtering for establishing master/detail relationship and does not refer to the server. Otherwise detail dataset performs query each time a record is selected in master dataset. This option is useful for reducing server calls number, server resources economy. It can be useful for slow connection. The [TMemDataSet.CachedUpdates](#) mode can be used for detail dataset only when this option is set to true. Setting the LocalMasterDetail option to True is not recommended when detail table contains too many rows, because when it is set to False, only records that correspond to the current record in master dataset are fetched.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.9.2.9 LongStrings Property

Used to represent string fields with the length that is greater than 255 as TStringField.

**Class**[TDADatasetOptions](#)**Syntax****property** LongStrings: boolean **default** True;**Remarks**

Use the LongStrings property to represent string fields with the length that is greater than 255 as TStringField, not as TMemoField.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.9.2.10 NumberRange Property

Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.

**Class**[TDADatasetOptions](#)**Syntax****property** NumberRange: boolean **default** False;**Remarks**

Use the NumberRange property to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.9.2.11 QueryRecCount Property

Used for TCustomDADataset to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.

**Class**[TDADatasetOptions](#)**Syntax****property** QueryRecCount: boolean **default** False;**Remarks**

If True, and the P:Devart.IbDac.TCustomIBCDataset.FetchAll property is False, TCustomDADataset performs additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records. Does not have any effect if the FetchAll property is True.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.9.2.12 QuoteNames Property

Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.

**Class**

[TDADatasetOptions](#)

**Syntax**

```
property QuoteNames: boolean default False;
```

**Remarks**

If True, TCustomDADataset quotes all database object names in autogenerated SQL statements such as update SQL.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.9.2.13 RemoveOnRefresh Property

Used for a dataset to locally remove a record that can not be found on the server.

**Class**

[TDADatasetOptions](#)

**Syntax**

```
property RemoveOnRefresh: boolean default True;
```

**Remarks**

When the RefreshRecord procedure can't find necessary record on the server and RemoveOnRefresh is set to True, dataset removes the record locally. Usually RefreshRecord can't find necessary record when someone else dropped the record or changed the key value of it.

This option makes sense only if the StrictUpdate option is set to False. If the StrictUpdate option is True, error will be generated regardless of the RemoveOnRefresh option value.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.9.2.14 RequiredFields Property

Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.

**Class**

[TDADatasetOptions](#)

**Syntax**

```
property RequiredFields: boolean default True;
```

**Remarks**

If True, TCustomDADataset sets the Required property of the TField objects for the NOT NULL fields. It is useful when table has a trigger which updates the NOT NULL



fields.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.9.2.15 ReturnParams Property

Used to return the new value of fields to dataset after insert or update.

### Class

[TDADatasetOptions](#)

### Syntax

```
property ReturnParams: boolean default False;
```

### Remarks

Use the ReturnParams property to return the new value of fields to dataset after insert or update. The actual value of field after insert or update may be different from the value stored in the local memory if the table has a trigger. When ReturnParams is True, OUT parameters of the SQLInsert and SQLUpdate statements is assigned to the corresponding fields.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.9.2.16 SetFieldsReadOnly Property

Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.

### Class

[TDADatasetOptions](#)

### Syntax

```
property SetFieldsReadOnly: boolean default True;
```

### Remarks

If True, dataset sets the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated. Set this option for datasets that use automatic generation of the update SQL statements only.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.9.2.17 StrictUpdate Property

Used for TCustomDADataset to raise an exception when the number of updated or deleted records is not equal 1.

### Class

[TDADatasetOptions](#)

### Syntax

```
property StrictUpdate: boolean default True;
```

**Remarks**

If True, TCustomDADataset raises an exception when the number of updated or deleted records is not equal 1. Setting this option also causes the exception if the RefreshRecord procedure returns more than one record. The exception does not occur when you execute SQL query, that doesn't return resultset.

**Note:** There can be problems if this option is set to True and triggers for UPDATE, DELETE, REFRESH commands that are defined for the table. So it is recommended to disable (set to False) this option with triggers.

TrimFixedChar specifies whether to discard all trailing spaces in the string fields of a dataset.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.9.2.18 TrimFixedChar Property

Specifies whether to discard all trailing spaces in the string fields of a dataset.

**Class**

[TDADatasetOptions](#)

**Syntax**

```
property TrimFixedChar: boolean default True;
```

**Remarks**

Specifies whether to discard all trailing spaces in the string fields of a dataset.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.9.2.19 UpdateAllFields Property

Used to include all dataset fields in the generated UPDATE and INSERT statements.

**Class**

[TDADatasetOptions](#)

**Syntax**

```
property UpdateAllFields: boolean default False;
```

**Remarks**

If True, all dataset fields will be included in the generated UPDATE and INSERT statements. Unspecified fields will have NULL value in the INSERT statements. Otherwise, only updated fields will be included to the generated update statements.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.9.2.20 UpdateBatchSize Property

Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

**Class**

### [TDADatasetOptions](#)

#### Syntax

```
property UpdateBatchSize: Integer default 1;
```

#### Remarks

Use the UpdateBatchSize property to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch. Takes effect only when updating dataset in the [TMemDataSet.CachedUpdates](#) mode. The default value is 1.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.1 TDAEncryptionOptions Class

Used to specify the options of the data encryption in a dataset.  
For a list of all members of this type, see [TDAEncryptionOptions](#) members.

#### Unit

[DBAccess](#)

#### Syntax

```
TDAEncryptionOptions = class(TPersistent);
```

#### Remarks

Set the properties of Encryption to specify the options of the data encryption in a dataset.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.10.1 Members

[TDAEncryptionOptions](#) class overview.

#### Properties

Name	Description
<a href="#">Encryptor</a>	Used to specify the encryptor class that will perform the data encryption.
<a href="#">Fields</a>	Used to set field names for which encryption will be performed.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.10.2 Properties

Properties of the **TDAEncryptionOptions** class.  
For a complete list of the **TDAEncryptionOptions** class members, see the [TDAEncryptionOptions Members](#) topic.

#### Public

Name	Description
<a href="#">Encryptor</a>	Used to specify the encryptor class that will perform the data encryption.

**Published**

Name	Description
<a href="#">Fields</a>	Used to set field names for which encryption will be performed.

**See Also**

- [TDAEncryptionOptions Class](#)
  - [TDAEncryptionOptions Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.10.2.1 Encryptor Property

Used to specify the encryptor class that will perform the data encryption.

**Class**

[TDAEncryptionOptions](#)

**Syntax**

```
property Encryptor: TCREncryptor;
```

**Remarks**

Use the Encryptor property to specify the encryptor class that will perform the data encryption.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.10.2.2 Fields Property

Used to set field names for which encryption will be performed.

**Class**

[TDAEncryptionOptions](#)

**Syntax**

```
property Fields: string;
```

**Remarks**

Used to set field names for which encryption will be performed. Field names must be separated by semicolons.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.1 TDAMapRule Class

Class that formes rules for Data Type Mapping.  
For a list of all members of this type, see [TDAMapRule](#) members.

#### Unit

[DBAccess](#)

#### Syntax

```
TDAMapRule = class (TMapRule) ;
```

#### Remarks

Using properties of this class, it is possible to change parameter values of the specified rules from the TDAMapRules set.

#### Inheritance Hierarchy

[TMapRule](#)  
**TDAMapRule**

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.11.1 Members

[TDAMapRule](#) class overview.

#### Properties

Name	Description
<a href="#">DBLengthMax</a>	Maximum DB field length, until which the rule is applied.
<a href="#">DBLengthMin</a>	Minimum DB field length, starting from which the rule is applied.
<a href="#">DBScaleMax</a>	Maximum DB field scale, until which the rule is applied to the specified DB field.
<a href="#">DBScaleMin</a>	Minimum DB field Scale, starting from which the rule is applied to the specified DB field.
<a href="#">DBType</a>	DB field type, that the rule is applied to.
<a href="#">FieldLength</a>	The resultant field length in Delphi.
<a href="#">FieldName</a>	DataSet field name, for which the rule is applied.
<a href="#">FieldScale</a>	The resultant field Scale in Delphi.

[FieldType](#)

Delphi field type, that the specified DB type or DataSet field will be mapped to.

[IgnoreErrors](#)

Ignoring errors when converting data from DB to Delphi type.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.11.2 Properties

Properties of the **TDAMapRule** class.

For a complete list of the **TDAMapRule** class members, see the [TDAMapRule Members](#) topic.

#### Published

Name	Description
<a href="#">DBLengthMax</a>	Maximum DB field length, until which the rule is applied.
<a href="#">DBLengthMin</a>	Minimum DB field length, starting from which the rule is applied.
<a href="#">DBScaleMax</a>	Maximum DB field scale, until which the rule is applied to the specified DB field.
<a href="#">DBScaleMin</a>	Minimum DB field Scale, starting from which the rule is applied to the specified DB field.
<a href="#">DBType</a>	DB field type, that the rule is applied to.
<a href="#">FieldLength</a>	The resultant field length in Delphi.
<a href="#">FieldName</a>	DataSet field name, for which the rule is applied.
<a href="#">FieldScale</a>	The resultant field Scale in Delphi.
<a href="#">FieldType</a>	Delphi field type, that the specified DB type or DataSet field will be mapped to.
<a href="#">IgnoreErrors</a>	Ignoring errors when converting data from DB to Delphi type.

#### See Also

- [TDAMapRule Class](#)
- [TDAMapRule Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.11.2.1 DBLengthMax Property

Maximum DB field length, until which the rule is applied.

##### Class

[TDAMapRule](#)

##### Syntax

```
property DBLengthMax: Integer default rlAny;
```

##### Remarks

Setting maximum DB field length, until which the rule is applied to the specified DB field.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.11.2.2 DBLengthMin Property

Minimum DB field length, starting from which the rule is applied.

##### Class

[TDAMapRule](#)

##### Syntax

```
property DBLengthMin: Integer default rlAny;
```

##### Remarks

Setting minimum DB field length, starting from which the rule is applied to the specified DB field.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.11.2.3 DBScaleMax Property

Maximum DB field scale, until which the rule is applied to the specified DB field.

##### Class

[TDAMapRule](#)

##### Syntax

```
property DBScaleMax: Integer default rlAny;
```

##### Remarks

Setting maximum DB field scale, until which the rule is applied to the specified DB field.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.11.2.4 DBScaleMin Property

Minimum DB field Scale, starting from which the rule is applied to the specified DB field.

**Class**

[TDAMapRule](#)

**Syntax**

```
property DBScaleMin: Integer default rlAny;
```

**Remarks**

Setting minimum DB field Scale, starting from which the rule is applied to the specified DB field.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.11.2.5 DBType Property

DB field type, that the rule is applied to.

**Class**

[TDAMapRule](#)

**Syntax**

```
property DBType: Word default dtUnknown;
```

**Remarks**

Setting DB field type, that the rule is applied to. If the current rule is set for Connection, the rule will be applied to all fields of the specified type in all DataSets related to this Connection.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.11.2.6 FieldLength Property

The resultant field length in Delphi.

**Class**

[TDAMapRule](#)

**Syntax**

```
property FieldLength: Integer default rlAny;
```

**Remarks**

Setting the Delphi field length after conversion.

---

© 1997-2013 Devart. All Rights Reserved.



## 17.10.1.11.2.7 FieldName Property

DataSet field name, for which the rule is applied.

**Class**

[TDAMapRule](#)

**Syntax**

```
property FieldName: string;
```

**Remarks**

Specifies the DataSet field name, that the rule is applied to. If the current rule is set for Connection, the rule will be applied to all fields with such name in DataSets related to this Connection.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.11.2.8 FieldScale Property

The resultant field Scale in Delphi.

**Class**

[TDAMapRule](#)

**Syntax**

```
property FieldScale: Integer default rlAny;
```

**Remarks**

Setting the Delphi field Scale after conversion.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.11.2.9 FieldType Property

Delphi field type, that the specified DB type or DataSet field will be mapped to.

**Class**

[TDAMapRule](#)

**Syntax**

```
property FieldType: TFieldType default ftUnknown;
```

**Remarks**

Setting Delphi field type, that the specified DB type or DataSet field will be mapped to.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.11.2.10 IgnoreErrors Property

Ignoring errors when converting data from DB to Delphi type.

**Class**

[TDAMapRule](#)

**Syntax**

```
property IgnoreErrors: Boolean default False;
```

**Remarks**

Allows to ignore errors while data conversion in case if data or DB data format cannot be recorded to the specified Delphi field type. The default value is false.

---

© 1997-2013 Devart. All Rights Reserved.

**17.10.1.1:TDAMapRules Class**

Used for adding rules for DataSet fields mapping with both identifying by field name and by field type and Delphi field types.

For a list of all members of this type, see [TDAMapRules](#) members.

**Unit**

[DBAccess](#)

**Syntax**

```
TDAMapRules = class (TMapRules);
```

**Inheritance Hierarchy**

TMapRules  
**TDAMapRules**

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.12.1 Members

[TDAMapRules](#) class overview.

**Methods**

Name	Description
<a href="#">AddDBTypeRule</a>	Overloaded. Adding rules for mapping Database field types to Delphi field types.
<a href="#">AddFieldNameRule</a>	Overloaded. Adding rules for mapping named fields to Delphi field types and setting resultant length and scale for Delphi fields

[AddRule](#)

A unified method of adding rules for mapping a DataSet named field or DB field type with the specified length and scale to a field type with the specified length and scale in Delphi.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.12.2 Methods

Methods of the **TDAMapRules** class.

For a complete list of the **TDAMapRules** class members, see the [TDAMapRules Members](#) topic.

**Public**

Name	Description
<a href="#">AddDBTypeRule</a>	Overloaded. Adding rules for mapping Database field types to Delphi field types.
<a href="#">AddFieldNameRule</a>	Overloaded. Adding rules for mapping named fields to Delphi field types and setting resultant length and scale for Delphi fields
<a href="#">AddRule</a>	A unified method of adding rules for mapping a DataSet named field or DB field type with the specified length and scale to a field type with the specified length and scale in Delphi.

**See Also**

- [TDAMapRules Class](#)
- [TDAMapRules Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.12.2.1 AddDBTypeRule Method

Adding rules for mapping Database field types to Delphi field types.

**Class**

[TDAMapRules](#)

**Overload List**

Name	Description
------	-------------

[AddDBTypeRule\(DBType: Word;  
FieldType: TFieldType; IgnoreErrors:  
boolean\)](#)

Adding rules for mapping Database field types to Delphi field types.

[AddDBTypeRule\(DBType: Word;  
FieldType: TFieldType; FieldLength:  
Integer; IgnoreErrors: boolean\)](#)

Adding rules for mapping Database field types to Delphi field types with the specified Delphi field length.

[AddDBTypeRule\(DBType: Word;  
FieldType: TFieldType; FieldLength:  
Integer; FieldScale: Integer;  
IgnoreErrors: boolean\)](#)

Adding rules for mapping Database field types to Delphi field types with the specified resultant length and scale of Delphi field.

[AddDBTypeRule\(DBType: Word;  
DBLengthMin: Integer; DBLengthMax:  
Integer; FieldType: TFieldType;  
IgnoreErrors: boolean\)](#)

Adding rules for mapping Database field types to Delphi field types with the specified minimum and maximum length of DB fields, for which the specified conversion will be applied.

[AddDBTypeRule\(DBType: Word;  
DBLengthMin: Integer; DBLengthMax:  
Integer; FieldType: TFieldType;  
FieldLength: Integer; IgnoreErrors:  
boolean\)](#)

Adding rules for mapping Database field types to Delphi field types with the specified minimum and maximum length of DB fields, for which the specified conversion will be applied.

[AddDBTypeRule\(DBType: Word;  
DBLengthMin: Integer; DBLengthMax:  
Integer; DBScaleMin: Integer;  
DBScaleMax: Integer; FieldType:  
TFieldType; IgnoreErrors: boolean\)](#)

Adding rules for mapping Database field types to Delphi field types with the specified minimum and maximum length and scale of DB fields, for which the specified conversion will be applied, and with setting the resultant Delphi field length.

[AddDBTypeRule\(DBType: Word;  
DBLengthMin: Integer; DBLengthMax:  
Integer; DBScaleMin: Integer;  
DBScaleMax: Integer; FieldType:  
TFieldType; FieldLength: Integer;  
FieldScale: Integer; IgnoreErrors:  
boolean\)](#)

Adding rules for mapping Database field types to Delphi field types with the specified minimum and maximum length and scale of DB fields, for which the specified conversion will be applied, and with setting the resultant Delphi field length and scale.

© 1997-2013 Devart. All Rights Reserved.

Adding rules for mapping Database field types to Delphi field types.

## Class

[TDAMapRules](#)

## Syntax

```
procedure AddDBTypeRule(DBType: Word; FieldType: TFieldType;  
IgnoreErrors: boolean = False); overload
```

### Parameters

*DBType*

DB type

*FieldType*

Delphi field type

*IgnoreErrors*

Ignore data conversion errors. Default value is False.

**Remarks**

This method can be applied to all DB fields and Delphi fields, that support conversion between each other.

---

© 1997-2013 Devart. All Rights Reserved.

Adding rules for mapping Database field types to Delphi field types with the specified Delphi field length.

**Class**

[TDAMapRules](#)

**Syntax**

```
procedure AddDBTypeRule(DBType: Word; FieldType: TFieldType;  
    FieldLength: Integer; IgnoreErrors: boolean = False); overload
```

**Parameters***DBType*

DB type

*FieldType*

Delphi field type

*FieldLength*

Delphi field length

*IgnoreErrors*

Ignore data conversion errors. Default value is False.

**Remarks**

This method can be used for retrieving Delphi fields ftString, ftWideString, ftBytes, ftVarBytes.

---

© 1997-2013 Devart. All Rights Reserved.

Adding rules for mapping Database field types to Delphi field types with the specified resultant length and scale of Delphi field.

**Class**

[TDAMapRules](#)

**Syntax**

```
procedure AddDBTypeRule(DBType: Word; FieldType: TFieldType;  
    FieldLength: Integer; FieldScale: Integer; IgnoreErrors: boolean  
    = False); overload
```

**Parameters***DBType*

DB type

*FieldType*

Delphi field type

*FieldLength*

Delphi field length

*FieldScale*

Delphi field scale

*IgnoreErrors*

Ignore data conversion errors. Default value is False.

### Remarks

This method can be used for retrieving Delphi fields ftBCD and ftFMTBCD.

---

© 1997-2013 Devart. All Rights Reserved.

Adding rules for mapping Database field types to Delphi field types with the specified minimum and maximum length of DB fields, for which the specified conversion will be applied.

### Class

[TDAMapRules](#)

### Syntax

```
procedure AddDBTypeRule(DBType: Word; DBLengthMin: Integer;  
    DBLengthMax: Integer; FieldType: TFieldType; IgnoreErrors:  
    boolean = False); overload
```

#### Parameters

*DBType*

DB type

*DBLengthMin*

Minimum DB field length

*DBLengthMax*

Maximum DB field length

*FieldType*

Delphi field type

*IgnoreErrors*

Ignore data conversion errors. Default value is False.

### Remarks

This method can be applied for all DB text fields.

---

© 1997-2013 Devart. All Rights Reserved.

Adding rules for mapping Database field types to Delphi field types with the specified minimum and maximum length of DB fields, for which the specified conversion will be applied.

### Class

[TDAMapRules](#)

## Syntax

```
procedure AddDBTypeRule(DBType: Word; DBLengthMin: Integer;  
    DBLengthMax: Integer; FieldType: TFieldType; FieldLength:  
    Integer; IgnoreErrors: boolean = False); overload
```

### Parameters

*DBType*

DB type

*DBLengthMin*

Minimum DB field length

*DBLengthMax*

Maximum DB field length

*FieldType*

Delphi field type

*FieldLength*

Delphi field length

*IgnoreErrors*

Ignore data conversion errors. Default value is False.

## Remarks

This method can be applied to DB text fields for retrieving Delphi fields `ftString`, `ftWideString`, `ftBytes`, `ftVarBytes`.

---

© 1997-2013 Devart. All Rights Reserved.

Adding rules for mapping Database field types to Delphi field types with the specified minimum and maximum length and scale of DB fields, for which the specified conversion will be applied, and with setting the resultant Delphi field length.

## Class

[TDAMapRules](#)

## Syntax

```
procedure AddDBTypeRule(DBType: Word; DBLengthMin: Integer;  
    DBLengthMax: Integer; DBScaleMin: Integer; DBScaleMax: Integer;  
    FieldType: TFieldType; IgnoreErrors: boolean = False); overload
```

### Parameters

*DBType*

DB type

*DBLengthMin*

Minimum DB field length

*DBLengthMax*

Maximum DB field length

*DBScaleMin*

Minimum DB field scale

*DBScaleMax*

Maximum DB field scale

*FieldType*

Delphi field type

*IgnoreErrors*

Ignore data conversion errors. Default value is False.

### Remarks

This method can be applied to those DB fields, for which it is possible to set Scale and Length.

---

© 1997-2013 Devart. All Rights Reserved.

Adding rules for mapping Database field types to Delphi field types with the specified minimum and maximum length and scale of DB fields, for which the specified conversion will be applied, and with setting the resultant Delphi field length and scale.

### Class

[TDAMapRules](#)

### Syntax

```
procedure AddDBTypeRule(DBType: Word; DBLengthMin: Integer;  
    DBLengthMax: Integer; DBScaleMin: Integer; DBScaleMax: Integer;  
    FieldType: TFieldType; FieldLength: Integer; FieldScale:  
    Integer; IgnoreErrors: boolean = False); overload
```

#### Parameters

*DBType*

DB type

*DBLengthMin*

Minimum DB field length

*DBLengthMax*

Maximum DB field length

*DBScaleMin*

Minimum DB field scale

*DBScaleMax*

Maximum DB field scale

*FieldType*

Delphi field type

*FieldLength*

Delphi field length

*FieldScale*

Delphi field scale

*IgnoreErrors*

Ignore data conversion errors. Default value is False.

### Remarks

This method can be applied to those DB fields, for which it is possible to set Scale



and Length for retrieving Delphi fields ftBCD, ftFMTBCD.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.12.2.2 AddFieldNameRule Method

Adding rules for mapping named fields to Delphi field types and setting resultant length and scale for Delphi fields

### Class

[TDAMapRules](#)

### Overload List

Name	Description
<a href="#">AddFieldNameRule(FieldName: string; FieldType: TFieldType; IgnoreErrors: Boolean)</a>	Adding rules for mapping named fields to Delphi field types.
<a href="#">AddFieldNameRule(FieldName: string; FieldType: TFieldType; FieldLength: Integer; IgnoreErrors: Boolean)</a>	Adding rules for mapping named fields to Delphi field types and setting the length for Delphi fields.
<a href="#">AddFieldNameRule(FieldName: string; FieldType: TFieldType; FieldLength: Integer; FieldScale: Integer; IgnoreErrors: Boolean)</a>	Adding rules for mapping named fields to Delphi field types and setting the resultant length and scale for Delphi fields

© 1997-2013 Devart. All Rights Reserved.

Adding rules for mapping named fields to Delphi field types.

### Class

[TDAMapRules](#)

### Syntax

```
procedure AddFieldNameRule(FieldName: string; FieldType: TFieldType; IgnoreErrors: Boolean = False); overload
```

#### Parameters

*FieldName*

Field name in DataSet

*FieldType*

Delphi field type

*IgnoreErrors*

Ignore data conversion errors. Default value is False.

### Remarks

This method can be applied to all DataSet field names and Delphi fields. If the DB field type, whose name is specified in the rule, doesn't support conversion to the specified Delphi type, the [Unsupported Data Type Mapping](#) error will occur when opening DataSet.

© 1997-2013 Devart. All Rights Reserved.

Adding rules for mapping named fields to Delphi field types and setting the length for Delphi fields.

## Class

[TDAMapRules](#)

## Syntax

```
procedure AddFieldNameRule(Fieldname: string; FieldType:
TFieldType; FieldLength: Integer; IgnoreErrors: Boolean =
False); overload
```

### Parameters

*FieldName*

Field name in DataSet

*FieldType*

Delphi field type

*FieldLength*

Delphi field length

*IgnoreErrors*

Ignore data conversion errors. Default value is False.

## Remarks

This method can be used for retrieving Delphi fields ftString, ftWideString, ftBytes, ftVarBytes.

---

© 1997-2013 Devart. All Rights Reserved.

Adding rules for mapping named fields to Delphi field types and setting the resultant length and scale for Delphi fields

## Class

[TDAMapRules](#)

## Syntax

```
procedure AddFieldNameRule(Fieldname: string; FieldType:
TFieldType; FieldLength: Integer; FieldScale: Integer;
IgnoreErrors: Boolean = False); overload
```

### Parameters

*FieldName*

Field name in DataSet

*FieldType*

Delphi field type

*FieldLength*

Delphi field length

*FieldScale*

Delphi field scale

### *IgnoreErrors*

Ignore data conversion errors. Default value is False.

## Remarks

This method can be used for retrieving Delphi fields ftBCD and ftFMTBCD.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.12.2.3 AddRule Method

A unified method of adding rules for mapping a DataSet named field or DB field type with the specified length and scale to a field type with the specified length and scale in Delphi.

## Class

[TDAMapRules](#)

## Syntax

```
procedure AddRule(Fieldname: string; DBType: Word; DBLengthMin: Integer; DBLengthMax: Integer; DBScaleMin: Integer; DBScaleMax: Integer; FieldType: TFieldType; FieldLength: Integer; FieldScale: Integer; IgnoreErrors: boolean = False); overload;  
procedure AddRule(Rule: string); overload;
```

### Parameters

#### *Fieldname*

Field name in DataSet

#### *DBType*

DB type

#### *DBLengthMin*

Minimum DB field length

#### *DBLengthMax*

Maximum DB field length

#### *DBScaleMin*

Minimum DB field scale

#### *DBScaleMax*

Maximum DB field scale

#### *FieldType*

Delphi field type

#### *FieldLength*

Delphi field length

#### *FieldScale*

Delphi field scale

#### *IgnoreErrors*

Ignore data conversion errors. Default value is False.

## Remarks

One of two parameters requires to be specified: Fieldname or DBType. Also, it is required to specify the FieldType parameter. The other parameters are not required,

therefore it is allowed to set the rAny constant for them instead of a specific value. If the rAny constant is set, then the given rule will be applied for all fields independently on their length and scale.

For example, if it is necessary to set the field length in a database to 20 or more, then DBLengthMin should be set to 20, and DBLengthMax - to rAny.

If it is necessary to set scale to 5 or less, then DBScaleMin should be set to rAny, and DBScaleMax - to 5.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.1:TDAMetaData Class

A class for retrieving metainformation of the specified database objects in the form of dataset.

For a list of all members of this type, see [TDAMetaData](#) members.

#### Unit

[DBAccess](#)

#### Syntax

```
TDAMetaData = class (TMemDataSet) ;
```

#### Remarks

TDAMetaData is a TDataSet descendant standing for retrieving metainformation of the specified database objects in the form of dataset. First of all you need to specify which kind of metainformation you want to see. For this you need to assign the [TDAMetaData.MetaDataKind](#) property. Provide one or more conditions in the [TDAMetaData.Restrictions](#) property to diminish the size of the resultset and get only information you are interested in.

Use the [TDAMetaData.GetMetaDataKinds](#) method to get the full list of supported kinds of meta data. With the [TDAMetaData.GetRestrictions](#) method you can find out what restrictions are applicable to the specified MetaDataKind.

#### Example

The code below demonstrates how to get information about columns of the 'emp' table:

```
MetaData.Connection := Connection;
MetaData.MetaDataKind := 'Columns';
MetaData.Restrictions.Values['TABLE NAME'] := 'Emp';
MetaData.Open;
```

#### Inheritance Hierarchy

[TMemDataSet](#)

**TDAMetaData**

#### See Also

- 

[TDAMetaData.MetaDataKind](#)

- [TDAMetaData.Restrictions](#)
- [TDAMetaData.GetMetaDataKinds](#)
- [TDAMetaData.GetRestrictions](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.13.1 Members

[TDAMetaData](#) class overview.

### Properties

Name	Description
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">Connection</a>	Used to specify a connection object to use to connect to a data store.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">MetaDataKind</a>	Used to specify which kind of metainformation to show.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">Restrictions</a>	Used to provide one or more conditions restricting the list of objects to be described.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.

### Methods

Name	Description
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.

<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">GetBlob</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">GetMetaDataKinds</a>	Used to get values acceptable in the MetaDataKind property.
<a href="#">GetRestrictions</a>	Used to find out which restrictions are applicable to a certain MetaDataKind.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">Prepare</a> (inherited from <a href="#">TMemDataSet</a> )	Allocates resources and creates field components for a dataset.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Indicates the current update status for the dataset when cached updates are enabled.

## Events

Name	Description
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.13.2 Properties

Properties of the **TDAMetaData** class.

For a complete list of the **TDAMetaData** class members, see the [TDAMetaData Members](#) topic.

## Public

Name	Description
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">Connection</a>	Used to specify a connection object to use to connect to a data store.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">GetBlob</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.

<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">MetaDataKind</a>	Used to specify which kind of metainformation to show.
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.
<a href="#">Prepare</a> (inherited from <a href="#">TMemDataSet</a> )	Allocates resources and creates field components for a dataset.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Restrictions</a>	Used to provide one or more conditions restricting the list of objects to be described.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.



[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

Used to indicate the update status for the current record when cached updates are enabled.

[UpdateResult](#) (inherited from [TMemDataSet](#))

Reads the status of the latest call to the `ApplyUpdates` method while cached updates are enabled.

[UpdatesPending](#) (inherited from [TMemDataSet](#))

Used to check the status of the cached updates buffer.

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Indicates the current update status for the dataset when cached updates are enabled.

### See Also

- [TDAMetaData Class](#)
- [TDAMetaData Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.13.2.1 Connection Property

Used to specify a connection object to use to connect to a data store.

### Class

[TDAMetaData](#)

### Syntax

```
property Connection: TCustomDAConnection;
```

### Remarks

Use the Connection property to specify a connection object to use to connect to a data store.  
Set at design-time by selecting from the list of provided `TCustomDAConnection` or its descendant class objects.  
At runtime, set the Connection property to reference an instantiated `TCustomDAConnection` object.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.13.2.2 MetaDataKind Property

Used to specify which kind of metainformation to show.

### Class

[TDAMetaData](#)

### Syntax

```
property MetaDataKind: string;
```

### Remarks

This string property specifies which kind of metainformation to show. The value of this property should be assigned before activating the component. If `MetaDataKind` equals to an empty string (the default value), the full value list that this property accepts will be shown.

They are described in the table below:

<b>MetaDataKind</b>	<b>Description</b>
Columns	show metainformation about columns of existing tables
Constraints	show metainformation about the constraints defined in the database
IndexColumns	show metainformation about indexed columns
Indexes	show metainformation about indexes in a database
MetaDataKinds	show the acceptable values of this property. You will get the same result if the <code>MetadataKind</code> property is an empty string
ProcedureParameters	show metainformation about parameters of existing procedures
Procedures	show metainformation about existing procedures
Restrictions	generates a dataset that describes which <a href="#">restrictions</a> are applicable to each <code>MetaDataKind</code>
Tables	show metainformation about existing tables
Databases	show metainformation about existing databases

If you provide a value that equals neither of the values described in the table, an error will be raised.

## See Also

- [Restrictions](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.13.2.3 Restrictions Property

Used to provide one or more conditions restricting the list of objects to be described.

## Class

[TDAMetaData](#)

## Syntax

```
property Restrictions: _TStrings;
```

## Remarks

Use the Restriction list to provide one or more conditions restricting the list of objects to be described. To see the full list of restrictions and to which metadata kinds they are applicable, you should assign the `Restrictions` value to the `MetaDataKind` property and view the result.

## See Also

- [MetadataKind](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.13.3 Methods

Methods of the **TDAMetaData** class.

For a complete list of the **TDAMetaData** class members, see the [TDAMetaData Members](#) topic.

## Public

Name	Description
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">GetBlob</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">GetMetaDataKinds</a>	Used to get values acceptable in the MetadataKind property.
<a href="#">GetRestrictions</a>	Used to find out which restrictions are applicable to a certain MetadataKind.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.

<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.
<a href="#">Prepare</a> (inherited from <a href="#">TMemDataSet</a> )	Allocates resources and creates field components for a dataset.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.

## See Also

- [TDAMetaData Class](#)
- [TDAMetaData Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.13.3.1 GetMetaDataKinds Method

Used to get values acceptable in the MetaDataKind property.

## Class

[TDAMetaData](#)

## Syntax

```
procedure GetMetaDataKinds(List: _TStrings);
```

### Parameters

*List*

Holds the object that will be filled with metadata kinds (restrictions).

## Remarks

Call the GetMetaDataKinds method to get values acceptable in the MetaDataKind property. The List parameter will be cleared and then filled with values.

## See Also

- [MetaDataKind](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.13.3.2 GetRestrictions Method

Used to find out which restrictions are applicable to a certain MetaDataKind.

## Class

[TDAMetaData](#)

## Syntax

```
procedure GetRestrictions(List: _TStrings; const MetaDataKind:  
string);
```

### Parameters

*List*

Holds the object that will be filled with metadata kinds (restrictions).

*MetaDataKind*

Holds the metadata kind for which restrictions are returned.

## Remarks

Call the GetRestrictions method to find out which restrictions are applicable to a certain MetaDataKind. The List parameter will be cleared and then filled with values.

## See Also

- [Restrictions](#)
  - [GetMetaDataKinds](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.1 TDAParam Class

A class that forms objects to represent the values of the [parameters set](#). For a list of all members of this type, see [TDAParam](#) members.

## Unit

[DBAccess](#)

## Syntax

```
TDAParam = class (TParam) ;
```

## Remarks

Use the properties of TDAParam to set the value of a parameter. Objects that use parameters create TDAParam objects to represent these parameters. For example, TDAParam objects are used by TCustomDASQL, TCustomDADataset. TDAParam shares many properties with TField, as both describe the value of a field in a dataset. However, a TField object has several properties to describe the field binding and the way the field is displayed, edited, or calculated, that are not needed in a TDAParam object. Conversely, TDAParam includes properties that indicate how the field value is passed as a parameter.

## See Also

- [TCustomDADataset](#)
  - [TCustomDASQL](#)
  - [TDAParams](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.1 Members

[TDAParam](#) class overview.

## Properties

Name	Description
<a href="#">AsBlob</a>	Used to set and read the value of the BLOB parameter as string.
<a href="#">AsBlobRef</a>	Used to set and read the value of the BLOB parameter as a TBlob object.

<a href="#">AsFloat</a>	Used to assign the value for a float field to a parameter.
<a href="#">AsInteger</a>	Used to assign the value for an integer field to the parameter.
<a href="#">AsLargeInt</a>	Used to assign the value for a LargeInteger field to the parameter.
<a href="#">AsMemo</a>	Used to assign the value for a memo field to the parameter.
<a href="#">AsMemoRef</a>	Used to set and read the value of the memo parameter as a TBlob object.
<a href="#">AsSQLTimeStamp</a>	Used to specify the value of the parameter when it represents a SQL timestamp field.
<a href="#">AsString</a>	Used to assign the string value to the parameter.
<a href="#">AsWideString</a>	Used to assign the Unicode string value to the parameter.
<a href="#">DataType</a>	Indicates the data type of the parameter.
<a href="#">IsNull</a>	Used to indicate whether the value assigned to a parameter is NULL.
<a href="#">ParamType</a>	Used to indicate the type of use for a parameter.
<a href="#">Size</a>	Specifies the size of a string type parameter.
<a href="#">Value</a>	Used to represent the value of the parameter as Variant.

## Methods

Name	Description
<a href="#">AssignField</a>	Assigns field name and field value to a param.
<a href="#">AssignFieldValue</a>	Assigns the specified field properties and value to a parameter.
<a href="#">LoadFromFile</a>	Places the content of a specified file into a TDAParam object.
<a href="#">LoadFromStream</a>	Places the content from a stream into a TDAParam object.

[SetBlobData](#)

Overloaded. Writes the data from a specified buffer to BLOB.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.2 Properties

Properties of the **TDAParam** class.

For a complete list of the **TDAParam** class members, see the [TDAParam Members](#) topic.

#### Public

Name	Description
<a href="#">AsBlob</a>	Used to set and read the value of the BLOB parameter as string.
<a href="#">AsBlobRef</a>	Used to set and read the value of the BLOB parameter as a TBlob object.
<a href="#">AsFloat</a>	Used to assign the value for a float field to a parameter.
<a href="#">AsInteger</a>	Used to assign the value for an integer field to the parameter.
<a href="#">AsLargeInt</a>	Used to assign the value for a LargeInteger field to the parameter.
<a href="#">AsMemo</a>	Used to assign the value for a memo field to the parameter.
<a href="#">AsMemoRef</a>	Used to set and read the value of the memo parameter as a TBlob object.
<a href="#">AsSQLTimeStamp</a>	Used to specify the value of the parameter when it represents a SQL timestamp field.
<a href="#">AsString</a>	Used to assign the string value to the parameter.
<a href="#">AsWideString</a>	Used to assign the Unicode string value to the parameter.
<a href="#">IsNull</a>	Used to indicate whether the value assigned to a parameter is NULL.

#### Published

Name	Description
------	-------------



[DataType](#)

Indicates the data type of the parameter.

[ParamType](#)

Used to indicate the type of use for a parameter.

[Size](#)

Specifies the size of a string type parameter.

[Value](#)

Used to represent the value of the parameter as Variant.

### See Also

- [TDAParam Class](#)
- [TDAParam Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.2.1 AsBlob Property

Used to set and read the value of the BLOB parameter as string.

### Class

[TDAParam](#)

### Syntax

```
property AsBlob: TBlobData;
```

### Remarks

Use the AsBlob property to set and read the value of the BLOB parameter as string. Setting AsBlob will set the DataType property to ftBlob. AsBlob is the value of the parameter when it represents the value of LONG RAW type.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.2.2 AsBlobRef Property

Used to set and read the value of the BLOB parameter as a TBlob object.

### Class

[TDAParam](#)

### Syntax

```
property AsBlobRef: TBlob;
```

### Remarks

Use the AsBlobRef property to set and read the value of the BLOB parameter as a TBlob object. Setting AsBlobRef will set the DataType property to ftBlob. Specifies the value of the parameter when it represents the value of LONG RAW type.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.14.2.3 AsFloat Property

Used to assign the value for a float field to a parameter.

**Class**

[TDAParam](#)

**Syntax**

```
property AsFloat: double;
```

**Remarks**

Use the AsFloat property to assign the value for a float field to the parameter.

Setting AsFloat will set the DataType property to dtFloat.

Read the AsFloat property to determine the value that was assigned to an output parameter, represented as Double. The value of the parameter will be converted to the Double value if possible.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.14.2.4 AsInteger Property

Used to assign the value for an integer field to the parameter.

**Class**

[TDAParam](#)

**Syntax**

```
property AsInteger: integer;
```

**Remarks**

Use the AsInteger property to assign the value for an integer field to the parameter.

Setting AsInteger will set the DataType property to dtInteger.

Read the AsInteger property to determine the value that was assigned to an output parameter, represented as a 32-bit integer. The value of the parameter will be converted to the Integer value if possible.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.14.2.5 AsLargeInt Property

Used to assign the value for a LargeInteger field to the parameter.

**Class**

[TDAParam](#)

**Syntax**

```
property AsLargeInt: Int64;
```

**Remarks**

Set the AsLargeInt property to assign the value for an Int64 field to the parameter.

Setting AsLargeInt will set the DataType property to dtLargeint.

Read the AsLargeInt property to determine the value that was assigned to an

output parameter, represented as a 64-bit integer. The value of the parameter will be converted to the Int64 value if possible.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.2.6 AsMemo Property

Used to assign the value for a memo field to the parameter.

##### Class

[TDAParam](#)

##### Syntax

```
property AsMemo: string;
```

##### Remarks

Use the AsMemo property to assign the value for a memo field to the parameter. Setting AsMemo will set the DataType property to ftMemo. AsMemo is the value of the parameter when it represents the value of LONG type.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.2.7 AsMemoRef Property

Used to set and read the value of the memo parameter as a TBlob object.

##### Class

[TDAParam](#)

##### Syntax

```
property AsMemoRef: TBlob;
```

##### Remarks

Use the AsMemoRef property to set and read the value of the memo parameter as a TBlob object. Setting AsMemoRef will set the DataType property to ftMemo. Specifies the value of the parameter when it represents the value of LONG type.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.2.8 AsSQLTimeStamp Property

Used to specify the value of the parameter when it represents a SQL timestamp field.

##### Class

[TDAParam](#)

##### Syntax

```
property AsSQLTimeStamp: TSQLTimeStamp;
```

##### Remarks

Set the AsSQLTimeStamp property to assign the value for a SQL timestamp field to

the parameter. Setting AsSQLTimeStamp sets the DataType property to ftTimeStamp.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.2.9 AsString Property

Used to assign the string value to the parameter.

#### Class

[TDAParam](#)

#### Syntax

```
property AsString: string;
```

#### Remarks

Use the AsString property to assign the string value to the parameter. Setting AsString will set the DataType property to ftString.

Read the AsString property to determine the value that was assigned to an output parameter represented as a string. The value of the parameter will be converted to a string.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.2.10 AsWideString Property

Used to assign the Unicode string value to the parameter.

#### Class

[TDAParam](#)

#### Syntax

```
property AsWideString: string;
```

#### Remarks

Set AsWideString to assign the Unicode string value to the parameter. Setting AsWideString will set the DataType property to ftWideString.

Read the AsWideString property to determine the value that was assigned to an output parameter, represented as a Unicode string. The value of the parameter will be converted to a Unicode string.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.2.11 DataType Property

Indicates the data type of the parameter.

#### Class

[TDAParam](#)

#### Syntax

```
property DataType: TFieldType stored IsDataTypeStored;
```

## Remarks

DataType is set automatically when a value is assigned to a parameter. Do not set DataType for bound fields, as this may cause the assigned value to be misinterpreted.

Read DataType to learn the type of data that was assigned to the parameter. Every possible value of DataType corresponds to the type of a database field.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.14.2.12 IsNull Property

Used to indicate whether the value assigned to a parameter is NULL.

## Class

[TDAParam](#)

## Syntax

```
property IsNull: boolean;
```

## Remarks

Use the IsNull property to indicate whether the value assigned to a parameter is NULL.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.14.2.13 ParamType Property

Used to indicate the type of use for a parameter.

## Class

[TDAParam](#)

## Syntax

```
property ParamType default DB . ptUnknown;
```

## Remarks

Objects that use TDAParam objects to represent field parameters set ParamType to indicate the type of use for a parameter.

To learn the description of TParamType refer to Delphi Help.

**Note:** The value of ParamType is important for LONG, LONG RAW, BLOB and CLOB parameters. To write data to database, set ptInput to ParamType, to read data from database, set ptOutput to ParamType.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.14.2.14 Size Property

Specifies the size of a string type parameter.

## Class

[TDAParam](#)

### Syntax

```
property Size: integer default 0;
```

### Remarks

Use the `Size` property to indicate the maximum number of characters the parameter may contain. Use the `Size` property only for Output parameters of the **ftString**, **ftFixedChar**, **ftBytes**, **ftVarBytes**, or **ftWideString** type.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.2.15 Value Property

Used to represent the value of the parameter as Variant.

### Class

[TDAParam](#)

### Syntax

```
property Value: variant stored IsValueStored;
```

### Remarks

The `Value` property represents the value of the parameter as Variant. Use `Value` in generic code that manipulates the values of parameters without the need to know the field type the parameter represent.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.3 Methods

Methods of the **TDAParam** class.

For a complete list of the **TDAParam** class members, see the [TDAParam Members](#) topic.

### Public

Name	Description
<a href="#">AssignField</a>	Assigns field name and field value to a param.
<a href="#">AssignFieldValue</a>	Assigns the specified field properties and value to a parameter.
<a href="#">LoadFromFile</a>	Places the content of a specified file into a TDAParam object.
<a href="#">LoadFromStream</a>	Places the content from a stream into a TDAParam object.
<a href="#">SetBlobData</a>	Overloaded. Writes the data from a specified buffer to BLOB.

---

**See Also**

- [TDAParam Class](#)
  - [TDAParam Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.3.1 AssignField Method

Assigns field name and field value to a param.

**Class**

[TDAParam](#)

**Syntax**

```
procedure AssignField(Field: TField);
```

**Parameters**

*Field*

Holds the field which name and value should be assigned to the param.

**Remarks**

Call the AssignField method to assign field name and field value to a param.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.14.3.2 AssignFieldValue Method

Assigns the specified field properties and value to a parameter.

**Class**

[TDAParam](#)

**Syntax**

```
procedure AssignFieldValue(Field: TField; const Value: Variant);  
virtual;
```

**Parameters**

*Field*

Holds the field the properties of which will be assigned to the parameter.

*Value*

Holds the value for the parameter.

**Remarks**

Call the AssignFieldValue method to assign the specified field properties and value to a parameter.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.14.3.3 LoadFromFile Method

Places the content of a specified file into a TDAParam object.

**Class**

[TDAParam](#)

**Syntax**

```
procedure LoadFromFile(const FileName: string; BlobType:  
TBlobType);
```

**Parameters***FileName*

Holds the name of the file.

*BlobType*

Holds a value that modifies the DataType property so that this TDAParam object now holds the BLOB value.

**Remarks**

Use the LoadFromFile method to place the content of a file specified by FileName into a TDAParam object. The BlobType value modifies the DataType property so that this TDAParam object now holds the BLOB value.

**See Also**

- [LoadFromStream](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.14.3.4 LoadFromStream Method

Places the content from a stream into a TDAParam object.

**Class**

[TDAParam](#)

**Syntax**

```
procedure LoadFromStream(Stream: TStream; BlobType: TBlobType);  
virtual;
```

**Parameters***Stream*

Holds the stream to copy content from.

*BlobType*

Holds a value that modifies the DataType property so that this TDAParam object now holds the BLOB value.

**Remarks**

Call the LoadFromStream method to place the content from a stream into a TDAParam object. The BlobType value modifies the DataType property so that this



TDAParam object now holds the BLOB value.

## See Also

- [LoadFromFile](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.14.3.5 SetBlobData Method

Writes the data from a specified buffer to BLOB.

## Class

[TDAParam](#)

## Overload List

Name	Description
<a href="#">SetBlobData(Buffer: TValueBuffer)</a>	Writes the data from a specified buffer to BLOB.
<a href="#">SetBlobData(Buffer: Pointer; Size: Integer)</a>	Writes the data from a specified buffer to BLOB.

© 1997-2013 Devart. All Rights Reserved.

Writes the data from a specified buffer to BLOB.

## Class

[TDAParam](#)

## Syntax

**procedure** SetBlobData(Buffer: TValueBuffer); **overload**  
**Parameters**

*Buffer*

Holds the pointer to the data.

© 1997-2013 Devart. All Rights Reserved.

Writes the data from a specified buffer to BLOB.

## Class

[TDAParam](#)

## Syntax

**procedure** SetBlobData(Buffer: Pointer; Size: Integer); **overload**  
**Parameters**

*Buffer*

Holds the pointer to data.

*Size*

Holds the number of bytes to read from the buffer.

### Remarks

Call the SetBlobData method to write data from a specified buffer to BLOB.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.1 TDAParams Class

This class is used to manage a list of TDAParam objects for an object that uses field parameters.

For a list of all members of this type, see [TDAParams](#) members.

### Unit

[DBAccess](#)

### Syntax

```
TDAParams = class(TParams);
```

### Remarks

Use TDAParams to manage a list of TDAParam objects for an object that uses field parameters. For example, TCustomDADataset objects and TCustomDASQL objects use TDAParams objects to create and access their parameters.

### See Also

- [TCustomDADataset.Params](#)
  - [TCustomDASQL.Params](#)
  - [TDAParam](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.15.1 Members

[TDAParams](#) class overview.

### Properties

Name	Description
<a href="#">Items</a>	Used to iterate through all parameters.

### Methods

Name	Description
<a href="#">FindParam</a>	Searches for a parameter with the specified name.
<a href="#">ParamByName</a>	Searches for a parameter with the specified name.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.15.2 Properties

Properties of the **TDAParams** class.

For a complete list of the **TDAParams** class members, see the [TDAParams Members](#) topic.

**Public**

Name	Description
<a href="#">Items</a>	Used to iterate through all parameters.

**See Also**

- [TDAParams Class](#)
- [TDAParams Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.15.2.1 Items Property(Indexer)

Used to iterate through all parameters.

**Class**

[TDAParams](#)

**Syntax**

```
property Items[Index: integer]: TDAParam; default;  
Parameters
```

*Index*

Holds an index in the range 0..Count - 1.

**Remarks**

Use the Items property to iterate through all parameters. Index identifies the index in the range 0..Count - 1. Items can reference a particular parameter by its index, but the ParamByName method is preferred in order to avoid depending on the order of the parameters.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.15.3 Methods

Methods of the **TDAParams** class.

For a complete list of the **TDAParams** class members, see the [TDAParams Members](#) topic.

**Public**

Name	Description
<a href="#">FindParam</a>	Searches for a parameter with the specified name.
<a href="#">ParamByName</a>	Searches for a parameter with the specified name.

**See Also**

- [TDAParams Class](#)
  - [TDAParams Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.15.3.1 FindParam Method

Searches for a parameter with the specified name.

**Class**

[TDAParams](#)

**Syntax**

```
function FindParam(const Value: string): TDAParam;
```

**Parameters**

*Value*

Holds the parameter name.

**Return Value**

a parameter, if a match was found. Nil otherwise.

**Remarks**

Use the FindParam method to find a parameter with the name passed in Value. If a match is found, FindParam returns the parameter. Otherwise, it returns nil. Use this method rather than a direct reference to the Items property to avoid depending on the order of the entries.

To locate more than one parameter at a time by name, use the GetParamList method instead. To get only the value of a named parameter, use the ParamValues property.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.15.3.2 ParamByName Method

Searches for a parameter with the specified name.

**Class**

[TDAParams](#)

**Syntax**

```
function ParamByName(const Value: string): TDAParam;
```

**Parameters**

*Value*

Holds the parameter name.

**Return Value**

a parameter, if the match was found. otherwise an exception is raised.

## Remarks

Use the ParamByName method to find a parameter with the name passed in Value. If a match was found, ParamByName returns the parameter. Otherwise, an exception is raised. Use this method rather than a direct reference to the [Items](#) property to avoid depending on the order of the entries.  
To locate a parameter by name without raising an exception if the parameter is not found, use the FindParam method.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.1(TDATransaction Class

A base class that implements functionality for controlling transactions. For a list of all members of this type, see [TDATransaction](#) members.

## Unit

[DBAccess](#)

## Syntax

```
TDATransaction = class (TComponent);
```

## Remarks

TDATransaction is a base class for components implementing functionality for managing transactions.  
Do not create instances of TDATransaction. Use descendants of the TDATransaction class instead.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.16.1 Members

[TDATransaction](#) class overview.

## Properties

Name	Description
<a href="#">Active</a>	Used to determine if the transaction is active.
<a href="#">DefaultCloseAction</a>	Used to specify the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.

## Methods

Name	Description
<a href="#">Commit</a>	Commits the current transaction.

[Rollback](#)

Discards all modifications of data associated with the current transaction and ends the transaction.

[StartTransaction](#)

Begins a new transaction.

## Events

Name	Description
<a href="#">OnError</a>	Used to process errors that occur during executing a transaction.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.16.2 Properties

Properties of the **TDATransaction** class.

For a complete list of the **TDATransaction** class members, see the [TDATransaction Members](#) topic.

## Public

Name	Description
<a href="#">Active</a>	Used to determine if the transaction is active.
<a href="#">DefaultCloseAction</a>	Used to specify the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.

## See Also

- [TDATransaction Class](#)
  - [TDATransaction Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.16.2.1 Active Property

Used to determine if the transaction is active.

## Class

[TDATransaction](#)

## Syntax

```
property Active: boolean;
```

## Remarks

Indicates whether the transaction is active. This property is read-only.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.16.2 DefaultCloseAction Property

Used to specify the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.

### Class

[TDATransaction](#)

### Syntax

```
property DefaultCloseAction: TCRTransactionAction default
    taRollback;
```

### Remarks

Use DefaultCloseAction to specify the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.16.3 Methods

Methods of the **TDATransaction** class.

For a complete list of the **TDATransaction** class members, see the [TDATransaction Members](#) topic.

### Public

Name	Description
<a href="#">Commit</a>	Commits the current transaction.
<a href="#">Rollback</a>	Discards all modifications of data associated with the current transaction and ends the transaction.
<a href="#">StartTransaction</a>	Begins a new transaction.

### See Also

- [TDATransaction Class](#)
- [TDATransaction Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.16.3.1 Commit Method

Commits the current transaction.

### Class

[TDATransaction](#)

### Syntax

```
procedure Commit; virtual;
```

### Remarks

Call the Commit method to commit the current transaction. On commit server writes permanently all pending data updates associated with the current transaction to the database, and then finishes the transaction.

### See Also

- [Rollback](#)
  - [StartTransaction](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.16.3.2 Rollback Method

Discards all modifications of data associated with the current transaction and ends the transaction.

### Class

[TDATransaction](#)

### Syntax

```
procedure Rollback; virtual;
```

### Remarks

Call Rollback to cancel all data modifications made within the current transaction to the database server, and finish the transaction.

### See Also

- [Commit](#)
  - [StartTransaction](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.16.3.3 StartTransaction Method

Begins a new transaction.

### Class

[TDATransaction](#)

### Syntax

```
procedure StartTransaction; virtual;
```

### Remarks

Call the StartTransaction method to begin a new transaction against the database server. Before calling StartTransaction, an application should check the [Active](#)



property. If `TDATransaction.Active` is `True`, indicating that a transaction is already in progress, a subsequent call to `StartTransaction` will raise `EDatabaseError`. An active transaction must be finished by call to [Commit](#) or [Rollback](#) before call to `StartTransaction`. Call to `StartTransaction` when connection is closed also will raise `EDatabaseError`.

Updates, insertions, and deletions that take place after a call to `StartTransaction` are held by the server until the application calls [Commit](#) to save the changes, or [Rollback](#) to cancel them.

## See Also

- [Commit](#)
- [Rollback](#)
- [TIBCTransaction.Active](#)
- [TIBCTransaction.Commit](#)
- [TIBCTransaction.Rollback](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.16.4 Events

Events of the **TDATransaction** class.

For a complete list of the **TDATransaction** class members, see the [TDATransaction Members](#) topic.

## Public

Name	Description
<a href="#">OnError</a>	Used to process errors that occur during executing a transaction.

## See Also

- [TDATransaction Class](#)
- [TDATransaction Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.10.1.16.4.1 OnError Event

Used to process errors that occur during executing a transaction.

## Class

[TDATransaction](#)

## Syntax

```
property OnError: TDATransactionOnErrorEvent;
```

## Remarks

Add a handler to the `OnError` event to process errors that occur during executing a transaction control statements such as [Commit](#), [Rollback](#). Check the `E` parameter to

get the error code.

### See Also

- [Commit](#)
  - [Rollback](#)
  - [StartTransaction](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.1 TMacro Class

Object that represents the value of a macro.

For a list of all members of this type, see [TMacro](#) members.

### Unit

[DBAccess](#)

### Syntax

```
TMacro = class (TCollectionItem);
```

### Remarks

TMacro object represents the value of a macro. Macro is a variable that holds string value. You just insert **&** MacroName in a SQL query text and change the value of macro by the Macro property editor at design time or the Value property at run time. At the time of opening query macro is replaced by its value.

If by any reason it is not convenient for you to use the ' **&** ' symbol as a character of macro replacement, change the value of the MacroChar variable.

### See Also

- [TMacros](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.17.1 Members

[TMacro](#) class overview.

### Properties

Name	Description
<a href="#">Active</a>	Used to determine if the macro should be expanded.
<a href="#">AsDateTime</a>	Used to set the TDateTime value to a macro.
<a href="#">AsFloat</a>	Used to set the float value to a macro.
<a href="#">AsInteger</a>	Used to set the integer value to a macro.

[AsString](#)

Used to assign the string value to a macro.

[Name](#)

Used to identify a particular macro.

[Value](#)

Used to set the value to a macro.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.17.2 Properties

Properties of the **TMacro** class.

For a complete list of the **TMacro** class members, see the [TMacro Members](#) topic.

#### Public

Name	Description
<a href="#">AsDateTime</a>	Used to set the TDateTime value to a macro.
<a href="#">AsFloat</a>	Used to set the float value to a macro.
<a href="#">AsInteger</a>	Used to set the integer value to a macro.
<a href="#">AsString</a>	Used to assign the string value to a macro.

#### Published

Name	Description
<a href="#">Active</a>	Used to determine if the macro should be expanded.
<a href="#">Name</a>	Used to identify a particular macro.
<a href="#">Value</a>	Used to set the value to a macro.

#### See Also

- [TMacro Class](#)
- [TMacro Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.17.2.1 Active Property

Used to determine if the macro should be expanded.

#### Class

[TMacro](#)

#### Syntax

```
property Active: boolean default True;
```

**Remarks**

When set to True, the macro will be expanded, otherwise macro definition is replaced by null string. You can use the Active property to modify the SQL property. The default value is True.

**Example**

```
IBCQuery.SQL.Text := 'SELECT * FROM Dept WHERE DeptNo > 20 &Cond1';  
IBCQuery.Macros[0].Value := 'and DName is NULL';  
IBCQuery.Macros[0].Active:= False;
```

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.17.2.2 AsDateTime Property

Used to set the TDateTime value to a macro.

**Class**

[TMacro](#)

**Syntax**

```
property AsDateTime: TDateTime;
```

**Remarks**

Use the AsDateTime property to set the TDateTime value to a macro.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.17.2.3 AsFloat Property

Used to set the float value to a macro.

**Class**

[TMacro](#)

**Syntax**

```
property AsFloat: double;
```

**Remarks**

Use the AsFloat property to set the float value to a macro.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.17.2.4 AsInteger Property

Used to set the integer value to a macro.

**Class**

[TMacro](#)

**Syntax**

```
property AsInteger: integer;
```

**Remarks**

Use the AsInteger property to set the integer value to a macro.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.17.2.5 AsString Property

Used to assign the string value to a macro.

**Class**

[TMacro](#)

**Syntax**

```
property AsString: string;
```

**Remarks**

Use the AsString property to assign the string value to a macro. Read the AsString property to determine the value of macro represented as a string.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.17.2.6 Name Property

Used to identify a particular macro.

**Class**

[TMacro](#)

**Syntax**

```
property Name: string;
```

**Remarks**

Use the Name property to identify a particular macro.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.17.2.7 Value Property

Used to set the value to a macro.

**Class**

[TMacro](#)

**Syntax**

```
property Value: string;
```

**Remarks**

Use the Value property to set the value to a macro.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.1 TMacros Class

Controls a list of TMacro objects for the [TCustomDASQL.Macros](#) or [TCustomDADataset](#) components.

For a list of all members of this type, see [TMacros](#) members.

#### Unit

[DBAccess](#)

#### Syntax

```
TMacros = class (TCollection);
```

#### Remarks

Use TMacros to manage a list of TMacro objects for the [TCustomDASQL](#) or [TCustomDADataset](#) components.

#### See Also

- [TMacro](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.18.1 Members

[TMacros](#) class overview.

#### Properties

Name	Description
<a href="#">Items</a>	Used to iterate through all the macros parameters.

#### Methods

Name	Description
<a href="#">AssignValues</a>	Copies the macros values and properties from the specified source.
Expand	Changes the macros in the passed SQL statement to their values.
<a href="#">FindMacro</a>	Searches for a TMacro object by its name.
<a href="#">IsEqual</a>	Compares itself with another TMacro object.
<a href="#">MacroByName</a>	Used to search for a macro with the specified name.

[Scan](#)

Creates a macros from the passed SQL statement.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.18.2 Properties

Properties of the **TMacros** class.

For a complete list of the **TMacros** class members, see the [TMacros Members](#) topic.

**Public**

Name	Description
<a href="#">Items</a>	Used to iterate through all the macros parameters.

**See Also**

- [TMacros Class](#)
- [TMacros Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.18.2.1 Items Property(Indexer)

Used to iterate through all the macros parameters.

**Class**

[TMacros](#)

**Syntax**

```
property Items[Index: integer]: TMacro; default;
```

**Parameters***Index*

Holds the index in the range 0..Count - 1.

**Remarks**

Use the Items property to iterate through all macros parameters. Index identifies the index in the range 0..Count - 1.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.18.3 Methods

Methods of the **TMacros** class.

For a complete list of the **TMacros** class members, see the [TMacros Members](#) topic.

**Public**

Name	Description
<a href="#">AssignValues</a>	Copies the macros values and properties from the specified source.

Expand

Changes the macros in the passed SQL statement to their values.

[FindMacro](#)

Searches for a TMacro object by its name.

[IsEqual](#)

Compares itself with another TMacro object.

[MacroByName](#)

Used to search for a macro with the specified name.

[Scan](#)

Creates a macros from the passed SQL statement.

### See Also

- [TMacros Class](#)
  - [TMacros Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.18.3.1 AssignValues Method

Copies the macros values and properties from the specified source.

### Class

[TMacros](#)

### Syntax

```
procedure AssignValues(Value: TMacros);
```

#### Parameters

*Value*

Holds the source to copy the macros values and properties from.

### Remarks

The Assign method copies the macros values and properties from the specified source. Macros are not recreated. Only the values of macros with matching names are assigned.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.18.3.2 FindMacro Method

Searches for a TMacro object by its name.

### Class

[TMacros](#)

### Syntax

```
function FindMacro(const Value: string): TMacro;
```

#### Parameters

*Value*

Holds the value of a macro to search for.

---



### Return Value

TMacro object if a match was found, nil otherwise.

### Remarks

Call the FindMacro method to find a macro with the name passed in Value. If a match is found, FindMacro returns the macro. Otherwise, it returns nil. Use this method rather than a direct reference to the Items property to avoid depending on the order of the entries.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.18.3.3 IsEqual Method

Compares itself with another TMacro object.

### Class

[TMacros](#)

### Syntax

```
function IsEqual(Value: TMacros): boolean;
```

#### Parameters

*Value*

Holds the values of TMacro objects.

#### Return Value

True, if the number of TMacro objects and the values of all TMacro objects are equal.

### Remarks

Call the IsEqual method to compare itself with another TMacro object. Returns True if the number of TMacro objects and the values of all TMacro objects are equal.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.18.3.4 MacroByName Method

Used to search for a macro with the specified name.

### Class

[TMacros](#)

### Syntax

```
function MacroByName(const Value: string): TMacro;
```

#### Parameters

*Value*

Holds a name of the macro to search for.

#### Return Value

TMacro object, if a macro with specified name was found.

**Remarks**

Call the MacroByName method to find a Macro with the name passed in Value. If a match is found, MacroByName returns the Macro. Otherwise, an exception is raised. Use this method rather than a direct reference to the Items property to avoid depending on the order of the entries.

To locate a macro by name without raising an exception if the parameter is not found, use the FindMacro method.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.18.3.5 Scan Method

Creates a macros from the passed SQL statement.

**Class**

[TMacros](#)

**Syntax**

```
procedure Scan(SQL: string);
```

**Parameters**

*SQL*

Holds the passed SQL statement.

**Remarks**

Call the Scan method to create a macros from the passed SQL statement. On that all existing TMacro objects are cleared.

---

© 1997-2013 Devart. All Rights Reserved.

**17.10.1.1!TPoolingOptions Class**

This class allows setting up the behaviour of the connection pool.  
For a list of all members of this type, see [TPoolingOptions](#) members.

**Unit**

[DBAccess](#)

**Syntax**

```
TPoolingOptions = class(TPersistent);
```

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.19.1 Members

[TPoolingOptions](#) class overview.

**Properties**

Name	Description
------	-------------

---

[ConnectionLifetime](#)

Used to specify the maximum time during which an opened connection can be used by connection pool.

[MaxPoolSize](#)

Used to specify the maximum number of connections that can be opened in connection pool.

[MinPoolSize](#)

Used to specify the minimum number of connections that can be opened in the connection pool.

[Validate](#)

Used for a connection to be validated when it is returned from the pool.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.19.2 Properties

Properties of the **TPoolingOptions** class.

For a complete list of the **TPoolingOptions** class members, see the [TPoolingOptions Members](#) topic.

### Published

Name	Description
<a href="#">ConnectionLifetime</a>	Used to specify the maximum time during which an opened connection can be used by connection pool.
<a href="#">MaxPoolSize</a>	Used to specify the maximum number of connections that can be opened in connection pool.
<a href="#">MinPoolSize</a>	Used to specify the minimum number of connections that can be opened in the connection pool.
<a href="#">Validate</a>	Used for a connection to be validated when it is returned from the pool.

### See Also

- [TPoolingOptions Class](#)
- [TPoolingOptions Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.19.2.1 ConnectionLifetime Property

Used to specify the maximum time during which an opened connection can be used by connection pool.

**Class**

[TPoolingOptions](#)

**Syntax**

```
property ConnectionLifetime: integer default 0;
```

**Remarks**

Use the ConnectionLifeTime property to specify the maximum time during which an opened connection can be used by connection pool. Measured in milliseconds. Pool deletes connections with exceeded connection lifetime when [TCustomDAConnection](#) is about to close. If the ConnectionLifetime property is set to 0 (by default), then the lifetime of connection is infinity. ConnectionLifetime concerns only inactive connections in the pool.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.19.2.2 MaxPoolSize Property

Used to specify the maximum number of connections that can be opened in connection pool.

**Class**

[TPoolingOptions](#)

**Syntax**

```
property MaxPoolSize: integer default 100;
```

**Remarks**

Specifies the maximum number of connections that can be opened in connection pool. Once this value is reached, no more connections are opened. The valid values are 1 and higher.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.10.1.19.2.3 MinPoolSize Property

Used to specify the minimum number of connections that can be opened in the connection pool.

**Class**

[TPoolingOptions](#)

**Syntax**

```
property MinPoolSize: integer default 0;
```

**Remarks**

Use the MinPoolSize property to specify the minimum number of connections that

can be opened in the connection pool.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.1.19.2.4 Validate Property

Used for a connection to be validated when it is returned from the pool.

### Class

[TPoolingOptions](#)

### Syntax

```
property Validate: boolean default False;
```

### Remarks

If the Validate property is set to True, connection will be validated when it is returned from the pool. By default this option is set to False and pool does not validate connection when it is returned to be used by a TCustomDAConnection component.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.2 Types

Types in the **DBAccess** unit.

### Types

Name	Description
<a href="#">TAfterExecuteEvent</a>	This type is used for the <a href="#">TCustomDADataSet.AfterExecute</a> and <a href="#">TCustomDASQL.AfterExecute</a> events.
<a href="#">TAfterFetchEvent</a>	This type is used for the <a href="#">TCustomDADataSet.AfterFetch</a> event.
<a href="#">TBeforeFetchEvent</a>	This type is used for the <a href="#">TCustomDADataSet.BeforeFetch</a> event.
<a href="#">TConnectionLostEvent</a>	This type is used for the <a href="#">TCustomDAConnection.OnConnectionLost</a> event.
<a href="#">TDAConnectionErrorEvent</a>	This type is used for the <a href="#">TCustomDAConnection.OnError</a> event.
<a href="#">TDATransactionErrorEvent</a>	This type is used for the <a href="#">TDATransaction.OnError</a> event.
<a href="#">TRefreshOptions</a>	Represents the set of <a href="#">TRefreshOption</a> .

[TUpdateExecuteEvent](#)

This type is used for the TCustomDADataset.AfterUpdateExecute and TCustomDADataset.BeforeUpdateExecute events.

© 1997-2013 Devart. All Rights Reserved.

**17.10.2.1 TAfterExecuteEvent Procedure Reference**

This type is used for the [TCustomDADataset.AfterExecute](#) and [TCustomDASQL.AfterExecute](#) events.

**Unit**

[DBAccess](#)

**Syntax**

```
TAfterExecuteEvent = procedure (Sender: TObject; Result: boolean)
of object;
```

**Parameters**

*Sender*

An object that raised the event.

*Result*

The result is True if SQL statement is executed successfully. False otherwise.

© 1997-2013 Devart. All Rights Reserved.

**17.10.2.2 TAfterFetchEvent Procedure Reference**

This type is used for the [TCustomDADataset.AfterFetch](#) event.

**Unit**

[DBAccess](#)

**Syntax**

```
TAfterFetchEvent = procedure (DataSet: TCustomDADataset) of object
;
```

**Parameters**

*DataSet*

Holds the TCustomDADataset descendant to synchronize the record position with.

© 1997-2013 Devart. All Rights Reserved.

**17.10.2.3 TBeforeFetchEvent Procedure Reference**

This type is used for the [TCustomDADataset.BeforeFetch](#) event.

**Unit**

[DBAccess](#)

## Syntax

```
TBeforeFetchEvent = procedure (DataSet: TCustomDADataSet; var
  Cancel: boolean) of object;
```

### Parameters

#### *DataSet*

Holds the TCustomDADataSet descendant to synchronize the record position with.

#### *Cancel*

True, if the current fetch operation should be aborted.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.2.4 TConnectionLostEvent Procedure Reference

This type is used for the [TCustomDAConnection.OnConnectionLost](#) event.

## Unit

[DBAccess](#)

## Syntax

```
TConnectionLostEvent = procedure (Sender: TObject; Component:
  TComponent; ConnLostCause: TConnLostCause; var RetryMode:
  TRetryMode) of object;
```

### Parameters

#### *Sender*

An object that raised the event.

#### *Component*

#### *ConnLostCause*

The reason of the connection loss.

#### *RetryMode*

The application behavior when connection is lost.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.2.5 TDAConnectionErrorEvent Procedure Reference

This type is used for the [TCustomDAConnection.OnError](#) event.

## Unit

[DBAccess](#)

## Syntax

```
TDAConnectionErrorEvent = procedure (Sender: TObject; E: EDAError;
  var Fail: boolean) of object;
```

### Parameters

#### *Sender*

An object that raised the event.

#### *E*

The error information.

*Fail*

False, if an error dialog should be prevented from being displayed and EAbort exception should be raised to cancel current operation .

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.2.6 TDATransactionErrorEvent Procedure Reference

This type is used for the [TDATransaction.OnError](#) event.

##### Unit

[DBAccess](#)

##### Syntax

```
TDATransactionErrorEvent = procedure (Sender: TObject; E: EDAEError  
; var Fail: boolean) of object;
```

##### Parameters

*Sender*

An object that raised the event.

*E*

The error code.

*Fail*

False, if an error dialog should be prevented from being displayed and EAbort exception to cancel the current operation should be raised.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.2.7 TRefreshOptions Set

Represents the set of [TRefreshOption](#).

##### Unit

[DBAccess](#)

##### Syntax

```
TRefreshOptions = set of TRefreshOption;
```

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.2.8 TUpdateExecuteEvent Procedure Reference

This type is used for the TCustomDADDataSet.AfterUpdateExecute and TCustomDADDataSet.BeforeUpdateExecute events.

##### Unit

[DBAccess](#)

##### Syntax

```
TUpdateExecuteEvent = procedure (Sender: TDataSet; StatementTypes:
```



```
TStatementTypes; Params: TDAParams) of object;
```

### Parameters

#### *Sender*

An object that raised the event.

#### *StatementTypes*

Holds the type of the SQL statement being executed.

#### *Params*

Holds the parameters with which the SQL statement will be executed.

© 1997-2013 Devart. All Rights Reserved.

## 17.10.3 Enumerations

Enumerations in the **DBAccess** unit.

### Enumerations

Name	Description
<a href="#">TLabelSet</a>	Sets the language of labels in the connect dialog.
<a href="#">TLockMode</a>	Specifies when to perform locking of an editing record.
<a href="#">TRefreshOption</a>	Indicates when the editing record will be refreshed.
<a href="#">TRetryMode</a>	Specifies the application behavior when connection is lost.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.3.1 TLabelSet Enumeration

Sets the language of labels in the connect dialog.

### Unit

[DBAccess](#)

### Syntax

```
TLabelSet = (lsCustom, lsEnglish, lsFrench, lsGerman, lsItalian,
lsPolish, lsPortuguese, lsRussian, lsSpanish);
```

### Values

Value	Meaning
<b>IsCustom</b>	Set the language of labels in the connect dialog manually.
<b>IsEnglish</b>	Set English as the language of labels in the connect dialog.
<b>IsFrench</b>	Set French as the language of labels in the connect dialog.
<b>IsGerman</b>	Set German as the language of labels in the connect dialog.
<b>IsItalian</b>	Set Italian as the language of labels in the connect dialog.
<b>IsPolish</b>	Set Polish as the language of labels in the connect dialog.

<b>IsPortuguese</b>	Set Portuguese as the language of labels in the connect dialog.
<b>IsRussian</b>	Set Russian as the language of labels in the connect dialog.
<b>IsSpanish</b>	Set Spanish as the language of labels in the connect dialog.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.3.2 TLockMode Enumeration

Specifies when to perform locking of an editing record.

##### Unit

[DBAccess](#)

##### Syntax

```
TLockMode = (lmNone);
```

##### Values

Value	Meaning
<b>ImLockDelayed</b>	Locking is performed when user posts an edited record. After this the lock is released. Corresponds to the optimistic locking. Locking is performed by the RefreshRecord method.
<b>ImLockImmediate</b>	Locking is performed when the user starts editing a record. The lock remains until the user posts or cancels the changes. Corresponds to the pessimistic locking.
<b>ImNone</b>	No locking is performed. This should only be used in single user applications. The default value.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.3.3 TRefreshOption Enumeration

Indicates when the editing record will be refreshed.

##### Unit

[DBAccess](#)

##### Syntax

```
TRefreshOption = (roAfterInsert, roAfterUpdate, roBeforeEdit);
```

##### Values

Value	Meaning
<b>roAfterInsert</b>	Refresh is performed after inserting.
<b>roAfterUpdate</b>	Refresh is performed after updating.
<b>roBeforeEdit</b>	Refresh is performed by Edit method.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.3.4 TRetryMode Enumeration

Specifies the application behavior when connection is lost.

#### Unit

[DBAccess](#)

#### Syntax

```
TRetryMode = (rmRaise, rmReconnect, rmReconnectExecute);
```

#### Values

Value	Meaning
<b>rmRaise</b>	An exception is raised.
<b>rmReconnect</b>	Reconnect is performed and then exception is raised.
<b>rmReconnectExecute</b>	Reconnect is performed and abortive operation is reexecuted. Exception is not raised.

© 1997-2013 Devart. All Rights Reserved.

### 17.10.4 Variables

Variables in the **DBAccess** unit.

#### Variables

Name	Description
<a href="#">ChangeCursor</a>	When set to True allows data access components to change screen cursor for the execution time.
<a href="#">MacroChar</a>	Determinates what character is used for macros.

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.4.1 ChangeCursor Variable

When set to True allows data access components to change screen cursor for the execution time.

#### Unit

[DBAccess](#)

#### Syntax

```
ChangeCursor: boolean;
```

© 1997-2013 Devart. All Rights Reserved.

#### 17.10.4.2 MacroChar Variable

Determinates what character is used for macros.

##### Unit

[DBAccess](#)

##### Syntax

```
MacroChar: _char;
```

---

© 1997-2013 Devart. All Rights Reserved.

### 17.11 Devart.Dac.DataAdapter

This unit contains implementation of the DADDataAdapter class.

#### Classes

Name	Description
<a href="#">DADDataAdapter</a>	DataAdapter serves as a bridge between a System.Data.DataSet and a TDataSet component (data source) for retrieving and saving data.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.11.1 Classes

Classes in the **Devart.Dac.DataAdapter** unit.

#### Classes

Name	Description
<a href="#">DADDataAdapter</a>	DataAdapter serves as a bridge between a System.Data.DataSet and a TDataSet component (data source) for retrieving and saving data.

---

© 1997-2013 Devart. All Rights Reserved.

##### 17.11.1.1 DADDataAdapter Class

DataAdapter serves as a bridge between a System.Data.DataSet and a TDataSet component (data source) for retrieving and saving data.

For a list of all members of this type, see [DADDataAdapter](#) members.

##### Unit

[Devart.Dac.DataAdapter](#)

##### Syntax

```
DADDataAdapter = class (TComponent) ;
```

## Remarks

DataAdapter serves as a bridge between a System.Data.DataSet and a TDataSet component (data source) for retrieving and saving data. DataAdapter provides this bridge by mapping [DADDataAdapter.Fill](#), which changes the data in the System.Data.DataSet to match the data in the data source, and [DADDataAdapter.Update](#), which changes the data in the data source to match the data in the System.Data.DataSet.

© 1997-2013 Devart. All Rights Reserved.

### 17.11.1.1.1 Members

[DADDataAdapter](#) class overview.

## Properties

Name	Description
<a href="#">DataSet</a>	Used to specify a TDataSet object which will be used as data source for DADDataAdapter component.

## Methods

Name	Description
<a href="#">Fill</a>	Adds or refreshes rows in the System.Data.DataSet to match those in the TDataSet and creates a DataTable.
<a href="#">Update</a>	Performs Insert, Edit, Delete for each inserted, updated, or deleted row in the specified System.Data.DataSet due to the ordering of the rows in the DataTable.

© 1997-2013 Devart. All Rights Reserved.

### 17.11.1.1.2 Properties

Properties of the **DADDataAdapter** class.

For a complete list of the **DADDataAdapter** class members, see the [DADDataAdapter Members](#) topic.

## Public

Name	Description
<a href="#">DataSet</a>	Used to specify a TDataSet object which will be used as data source for DADDataAdapter component.

**See Also**

- [DADDataAdapter Class](#)
  - [DADDataAdapter Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.11.1.1.2.1 DataSet Property

Used to specify a TDataSet object which will be used as data source for DADDataAdapter component.

**Class**

[DADDataAdapter](#)

**Syntax**

```
property DataSet: TDataSet;
```

**Remarks**

Specify a TDataSet object which will be used as data source for DADDataAdapter component.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.11.1.1.3 Methods

Methods of the **DADDataAdapter** class.

For a complete list of the **DADDataAdapter** class members, see the [DADDataAdapter Members](#) topic.

**Public**

Name	Description
<a href="#">Fill</a>	Adds or refreshes rows in the System.Data.DataSet to match those in the TDataSet and creates a DataTable.
<a href="#">Update</a>	Performs Insert, Edit, Delete for each inserted, updated, or deleted row in the specified System.Data.DataSet due to the ordering of the rows in the DataTable.

**See Also**

- [DADDataAdapter Class](#)
  - [DADDataAdapter Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.11.1.1.3.1 Fill Method

Adds or refreshes rows in the System.Data.DataSet to match those in the TDataSet and creates a DataTable.

**Class**

[DADDataAdapter](#)

**Syntax**

```
function Fill(Data: DataSet; tableName: string): integer;
```

**Parameters***Data*

holds the dataset updates of which are to be commented to the database.

*tableName*

holds the name of the DataTable.

**Return Value**

the number of rows successfully inserted into DataSet.

**Remarks**

Adds or refreshes rows in the System.Data.DataSet to match those in the TDataSet using the DataSet parameter, and creates a DataTable named tableName. Function returns the number of rows successfully inserted into DataSet.

TDataSet object associated with DADDataAdapter must be valid, but it does not need to be opened. If TDataSet is closed before Fill is called, it is opened to retrieve data, then closed. If TDataSet is opened before Fill is called, it remains opened.

If an error is encountered while populating the dataset, rows added prior to the occurrence of the error remain in the dataset. The remainder of the operation is aborted.

If TDataSet does not return any rows, fields are created and no rows are added to the DataSet, and no exception is raised.

**See Also**

- [Update](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.11.1.1.3.2 Update Method

Performs Insert, Edit, Delete for each inserted, updated, or deleted row in the specified System.Data.DataSet due to the ordering of the rows in the DataTable.

**Class**

[DADDataAdapter](#)

**Syntax**

```
function Update(Data: DataSet; tableName: string): integer;
```

**Parameters**

**Data**

holds the dataset updates of which are to be commented to the database.

**tableName**

holds the name of the DataTable.

**Return Value**

the number of rows successfully updated from the DataSet.

**Remarks**

Performs Insert, Edit, Delete for each inserted, updated, or deleted row in the specified System.Data.DataSet due to the ordering of the rows in the DataTable. It should be noted that these statements are not performed as a batch process; each row is updated individually. Function returns the number of rows successfully updated from the DataSet.

**See Also**

- [Fill](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.12 Devart.IbDac.DataAdapter

This unit contains implementation of the IBIDataAdapter class.

**Classes**

Name	Description
<a href="#">IBIDataAdapter</a>	A class for using with <a href="#">TCustomIBDataSet</a> components and as data source for retrieving and saving data.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.12.1 Classes

Classes in the **Devart.IbDac.DataAdapter** unit.

**Classes**

Name	Description
<a href="#">IBIDataAdapter</a>	A class for using with <a href="#">TCustomIBDataSet</a> components and as data source for retrieving and saving data.

---

© 1997-2013 Devart. All Rights Reserved.



### 17.12.1.1 IBCDataAdapter Class

A class for using with [TCustomIBCDataset](#) components and as data source for retrieving and saving data.

For a list of all members of this type, see [IBCDDataAdapter](#) members.

#### Unit

[Devart.IbDac.DataAdapter](#)

#### Syntax

```
IBCDDataAdapter = class(DADDataAdapter);
```

#### Remarks

The IBCDataAdapter class is designed for usage with [TCustomIBCDataset](#) components and as data source for retrieving and saving data. IBCDataAdapter provides this bridge by mapping [DADDataAdapter.Fill](#), which changes the data in the System.Data.DataSet to match the data in the data source, and [DADDataAdapter.Update](#), which changes the data in the data source to match the data in the System.Data.DataSet.

#### Inheritance Hierarchy

[DADDataAdapter](#)

**IBCDDataAdapter**

#### See Also

- [DADDataAdapter](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.12.1.1.1 Members

[IBCDDataAdapter](#) class overview.

#### Properties

Name	Description
<a href="#">DataSet</a> (inherited from <a href="#">DADDataAdapter</a> )	Used to specify a TDataSet object which will be used as data source for DADDataAdapter component.

#### Methods

Name	Description
<a href="#">Fill</a> (inherited from <a href="#">DADDataAdapter</a> )	Adds or refreshes rows in the System.Data.DataSet to match those in the TDataSet and creates a DataTable.

[Update](#) (inherited from [DADDataAdapter](#))

Performs Insert, Edit, Delete for each inserted, updated, or deleted row in the specified System.Data.DataSet due to the ordering of the rows in the DataTable.

© 1997-2013 Devart. All Rights Reserved.

## 17.13 IBC

This unit contains main components of IBDAC.

### Classes

Name	Description
<a href="#">TCustomIBCDataset</a>	A base class that defines InterBase functionality for a dataset.
<a href="#">TCustomIBCQuery</a>	A base class for defining functionality for descendant classes which access database using SQL statements. c
<a href="#">TCustomIBCTable</a>	A base class that defines functionality for descendant classes which access data in a single table without writing SQL statements.
<a href="#">TIBCArray</a>	A class representing the value of the InterBase array data type.
<a href="#">TIBCArrayField</a>	A class encapsulating the fundamental behavior common to the InterBase array fields.
<a href="#">TIBConnection</a>	A component for setting and controlling connections to an InterBase database.
<a href="#">TIBConnectionOptions</a>	This class allows setting up the behaviour of the TIBConnection class.
<a href="#">TIBCDatasetOptions</a>	This class allows setting up the behaviour of the TIBCDataset class.
<a href="#">TIBCDatasource</a>	TIBCDatasource provides an interface between an IBDAC dataset components and data-aware controls on a form.

<a href="#">TIBCDbKeyField</a>	A class representing the InterBase RDB\$DB KEY field.
<a href="#">TIBCEncryptor</a>	The class that performs encrypting and decrypting of data.
<a href="#">TIBCMetaData</a>	A component for obtaining metainformation about database objects from the server.
<a href="#">TIBCPParam</a>	Description of TIBCPParam is not available at the moment
<a href="#">TIBCPParams</a>	Used to control TIBCPParam objects.
<a href="#">TIBCQuery</a>	A component for executing queries and operating record sets. It also provides flexible way to update data.
<a href="#">TIBCSQL</a>	A component for executing SQL statements and calling stored procedures on the database server.
<a href="#">TIBCSStoredProc</a>	A component for accessing and executing stored procedures and functions.
<a href="#">TIBCTable</a>	A component for retrieving and updating data in a single table without writing SQL statements.
<a href="#">TIBCTransaction</a>	A component for managing transactions in an application.
<a href="#">TIBCUpdateSQL</a>	A component for tuning update operations for the DataSet component.

## Types

Name	Description
<a href="#">TIBCTransactionErrorEvent</a>	This type is used for the <a href="#">TIBCTransaction.OnError</a> event.

## Enumerations

Name	Description
<a href="#">TGeneratorMode</a>	Specifies the method used internally to generate a sequenced field.

[TIBCTransactionAction](#)

Specifies the network protocol of connection with InterBase server.

[TIBCTransactionAction](#)

Specifies the transaction behaviour when connection closes.

## Routines

Name	Description
<a href="#">DefaultConnection</a>	Call this function to get pointer to default connection object.

## Variables

Name	Description
<a href="#">Connections</a>	Holds pointers to all TIBCTransactionAction objects of an application.
<a href="#">DefConnection</a>	Read this variable to get pointer to default connection object. Same as DefaultConnection function.
<a href="#">UseDefConnection</a>	When set to true enables TCustomIBCTransactionAction and TIBCTransactionAction components to use default connection if they are not attached to any connection.

## Constants

Name	Description
<a href="#">IBDACVersion</a>	Read this constant to get current version number for IBDAC.

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1 Classes

Classes in the **IBC** unit.

## Classes

Name	Description
<a href="#">TCustomIBCTransactionAction</a>	A base class that defines InterBase functionality for a dataset.

[TCustomIBCQuery](#)

A base class for defining functionality for descendant classes which access database using SQL statements. c

[TCustomIBCTable](#)

A base class that defines functionality for descendant classes which access data in a single table without writing SQL statements.

[TIBCArray](#)

A class representing the value of the InterBase array data type.

[TIBCArrayField](#)

A class encapsulating the fundamental behavior common to the InterBase array fields.

[TIBConnection](#)

A component for setting and controlling connections to an InterBase database.

[TIBConnectionOptions](#)

This class allows setting up the behaviour of the TIBConnection class.

[TIBDataSetOptions](#)

This class allows setting up the behaviour of the TIBDataSet class.

[TIBDataSource](#)

TIBDataSource provides an interface between an IBDAC dataset components and data-aware controls on a form.

[TIBCDbKeyField](#)

A class representing the InterBase RDB\$DB KEY field.

[TIBCEncryptor](#)

The class that performs encrypting and decrypting of data.

[TIBCMetaData](#)

A component for obtaining metainformation about database objects from the server.

[TIBCPParam](#)[TIBCPParams](#)

Used to control TIBCPParam objects.

[TIBCQuery](#)

A component for executing queries and operating record sets. It also provides flexible way to update data.

[TIBCSQL](#)

A component for executing SQL statements and calling stored procedures on the database server.

[TIBCStoredProc](#)

A component for accessing and executing stored procedures and functions.

[TIBCTable](#)

A component for retrieving and updating data in a single table without writing SQL statements.

[TIBCTransaction](#)

A component for managing transactions in an application.

[TIBCUpdateSQL](#)

A component for tuning update operations for the DataSet component.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1 TCustomIBCDataset Class

A base class that defines InterBase functionality for a dataset.

For a list of all members of this type, see [TCustomIBCDataset](#) members.

#### Unit

[IBC](#)

#### Syntax

```
TCustomIBCDataset = class (TCustomDADataset) ;
```

#### Remarks

TCustomIBCDataset is a component that defines InterBase functionality for a dataset. TCustomIBCDataset can execute queries, fetch rows and controls InterBase specific data types. Applications never use TCustomIBCDataset objects directly. Instead they use the descendants of TCustomIBCDataset, such as TIBCQuery, TIBCStoredProc and TIBCTable, which inherit its database-related properties and methods.

TIBCQuery provides insert, delete and update operations on records by dynamically generated SQL statements. It uses [TCustomDADataset.KeyFields](#) property to build SQL statements for the SQLDelete, SQLInsert and SQLUpdate properties if they were empty before updating the database.

#### Inheritance Hierarchy

[TMemDataSet](#)

[TCustomDADataset](#)

**TCustomIBCDataset**

#### See Also

- [TCustomDADataset.KeyFields](#)
- [TIBCQuery](#)
- [TIBCStoredProc](#)
- [TIBCTable](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1.1 Members

[TCustomIBCDataset](#) class overview.

### Properties

Name	Description
<a href="#">AutoCommit</a>	Used to automatically commit each update, insert or delete statement by database server.
<a href="#">BaseSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataset</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">Connection</a>	Used to specify the connection in which the dataset will be executed.
<a href="#">Cursor</a>	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
<a href="#">Debug</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#">DetailFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#">Disconnected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to keep dataset opened after connection is closed.
<a href="#">DMLRefresh</a>	Used to refresh record by the RETURNIG clause when insert is performed.
<a href="#">Encryption</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the options of the data encryption in a dataset.

<a href="#">FetchRows</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to define the number of rows to be transferred across the network at the same time.
<a href="#">FilterSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#">FinalSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#">GeneratorMode</a>	Used to specify which method is used internally to generate a sequenced field.
<a href="#">GeneratorStep</a>	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
<a href="#">Handle</a>	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">IsQuery</a>	Used to check if the SQL statement returns rows.
<a href="#">KeyFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#">KeyGenerator</a>	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.



<a href="#">LockMode</a>	Used to indicate when to perform a locking of editing record.
<a href="#">MacroCount</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to get the number of macros associated with the Macros property.
<a href="#">Macros</a> (inherited from <a href="#">TCustomDADataset</a> )	Makes it possible to change SQL queries easily.
<a href="#">MasterFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#">MasterSource</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the data source component which binds current dataset to the master one.
<a href="#">Options</a>	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#">ParamCheck</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#">ParamCount</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate how many parameters are there in the Params property.
<a href="#">Params</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#">Plan</a>	Used to get or set the PLAN clause of the SELECT statement.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">ReadOnly</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#">RefreshOptions</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate when the editing record is refreshed.
<a href="#">RowsAffected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

[RowsDeleted](#)

Used to indicate the number of rows that were deleted during the last query operation.

[RowsFetched](#)

Used to get the number of the currently fetched rows.

[RowsInserted](#)

Used to indicate the number of rows that were inserted during the last query operation.

[RowsUpdated](#)

Used to indicate the number of rows that were updated during the last query operation.

[SQL](#) (inherited from [TCustomDADataset](#))

Used to provide a SQL statement that a query component executes when its Open method is called.

[SQLDelete](#) (inherited from [TCustomDADataset](#))

Used to specify a SQL statement that will be used when applying a deletion to a record.

[SQLInsert](#) (inherited from [TCustomDADataset](#))

Used to specify the SQL statement that will be used when applying an insertion to a dataset.

[SQLLock](#) (inherited from [TCustomDADataset](#))

Used to specify a SQL statement that will be used to perform a record lock.

[SQLRefresh](#) (inherited from [TCustomDADataset](#))

Used to specify a SQL statement that will be used to refresh current record by calling the [TCustomDADataset.RefreshRecord](#) procedure.

[SQLType](#)

Used to get the typecode of the SQL statement being processed by the InterBase database server.

[SQLUpdate](#) (inherited from [TCustomDADataset](#))

Used to specify a SQL statement that will be used when applying an update to a dataset.

[Transaction](#)

Used to determine the transaction under which the query of this dataset executes.

[UniDirectional](#) (inherited from [TCustomDADataset](#))

Used if an application does not need bidirectional access to records in the result set.

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

Used to indicate the update status for the current record when cached updates are enabled.

[UpdatesPending](#) (inherited from [TMemDataSet](#))

Used to check the status of the cached updates buffer.

[UpdateTransaction](#)

Used to get or set the transaction for modifying a dataset.

## Methods

Name	Description
<a href="#">AddWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">BreakExec</a> (inherited from <a href="#">TCustomDADataset</a> )	Breaks execution of the SQL statement on the server.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">CreateBlobStream</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#">CreateProcCall</a>	Assigns PL/SQL block that calls stored procedure to the SQL property.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">DeleteWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#">Execute</a> (inherited from <a href="#">TCustomDADataset</a> )	Executes a SQL statement on the server.
<a href="#">Executing</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether SQL statement is still being executed.
<a href="#">Fetched</a>	Used to determine if TCustomDADataset has already fetched all rows.

<a href="#">Fetching</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#">FetchingAll</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#">FindKey</a> (inherited from <a href="#">TCustomDADataset</a> )	Searches for a record which contains specified field values.
<a href="#">FindMacro</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether a specified macro exists in a dataset.
<a href="#">FindNearest</a> (inherited from <a href="#">TCustomDADataset</a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#">FindParam</a>	Determines if a parameter with the specified name exists in a dataset.
<a href="#">GetArray</a>	Retrieves a TIBCArrary object for a field when only its name is known.
<a href="#">GetBlob</a>	Retrieves a TIBCBlob object for a field when only its name is known.
<a href="#">GetDataType</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#">GetFieldObject</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns a multireference shared object from field.
<a href="#">GetFieldPrecision</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the precision of a number field.
<a href="#">GetFieldScale</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the scale of a number field.
<a href="#">GetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves an ORDER BY clause from a SQL statement.
<a href="#">GotoCurrent</a> (inherited from <a href="#">TCustomDADataset</a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.

<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">Lock</a> (inherited from <a href="#">TCustomDADataset</a> )	Locks the current record.
<a href="#">MacroByName</a> (inherited from <a href="#">TCustomDADataset</a> )	Finds a Macro with the name passed in Name.
<a href="#">ParamByName</a>	Called to set or use parameter information for a specific parameter based on its name.
<a href="#">Prepare</a> (inherited from <a href="#">TCustomDADataset</a> )	Allocates, opens, and parses cursor for a query.
<a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )	Actuali es field values for the current record.
<a href="#">RestoreSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.

[UpdateResult](#) (inherited from [TMemDataSet](#))

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Indicates the current update status for the dataset when cached updates are enabled.

## Events

Name	Description
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">BeforeFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#">BeforeUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1.2 Properties

Properties of the **TCustomIBCDataset** class.  
For a complete list of the **TCustomIBCDataset** class members, see the [TCustomIBCDataset Members](#) topic.

## Public

Name	Description
<a href="#">AddWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after a component has executed a query to database.

<a href="#">AfterFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">AutoCommit</a>	Used to automatically commit each update, insert or delete statement by database server.
<a href="#">BaseSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#">BeforeFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#">BeforeUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#">BreakExec</a> (inherited from <a href="#">TCustomDADataset</a> )	Breaks execution of the SQL statement on the server.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">Connection</a>	Used to specify the connection in which the dataset will be executed.
<a href="#">CreateBlobStream</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#">Cursor</a>	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.

<a href="#"><u>Debug</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#"><u>DeferredPost</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Makes permanent changes to the database server.
<a href="#"><u>DeleteWhere</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#"><u>DetailFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#"><u>Disconnected</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to keep dataset opened after connection is closed.
<a href="#"><u>DMLRefresh</u></a>	Used to refresh record by the RETURNIG clause when insert is performed.
<a href="#"><u>Encryption</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the options of the data encryption in a dataset.
<a href="#"><u>Execute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Executes a SQL statement on the server.
<a href="#"><u>Executing</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether SQL statement is still being executed.
<a href="#"><u>Fetches</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset has already fetched all rows.
<a href="#"><u>Fetching</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#"><u>FetchingAll</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#"><u>FetchRows</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to define the number of rows to be transferred across the network at the same time.
<a href="#"><u>FilterSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#"><u>FinalSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.



<a href="#">FindKey</a> (inherited from <a href="#">TCustomDADataset</a> )	Searches for a record which contains specified field values.
<a href="#">FindMacro</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether a specified macro exists in a dataset.
<a href="#">FindNearest</a> (inherited from <a href="#">TCustomDADataset</a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#">FindParam</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if a parameter with the specified name exists in a dataset.
<a href="#">GeneratorMode</a>	Used to specify which method is used internally to generate a sequenced field.
<a href="#">GeneratorStep</a>	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
<a href="#">GetBlob</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">GetDataType</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#">GetFieldObject</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns a multireference shared object from field.
<a href="#">GetFieldPrecision</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the precision of a number field.
<a href="#">GetFieldScale</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the scale of a number field.
<a href="#">GetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves an ORDER BY clause from a SQL statement.
<a href="#">GotoCurrent</a> (inherited from <a href="#">TCustomDADataset</a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#">Handle</a>	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.

<a href="#">IsQuery</a>	Used to check if the SQL statement returns rows.
<a href="#">KeyFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#">KeyGenerator</a>	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">Lock</a> (inherited from <a href="#">TCustomDADataset</a> )	Locks the current record.
<a href="#">LockMode</a>	Used to indicate when to perform a locking of editing record.
<a href="#">MacroByName</a> (inherited from <a href="#">TCustomDADataset</a> )	Finds a Macro with the name passed in Name.
<a href="#">MacroCount</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to get the number of macros associated with the Macros property.
<a href="#">Macros</a> (inherited from <a href="#">TCustomDADataset</a> )	Makes it possible to change SQL queries easily.
<a href="#">MasterFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.

<a href="#">MasterSource</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the data source component which binds current dataset to the master one.
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.
<a href="#">Options</a>	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#">ParamByName</a> (inherited from <a href="#">TCustomDADataset</a> )	Sets or uses parameter information for a specific parameter based on its name.
<a href="#">ParamCheck</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#">ParamCount</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate how many parameters are there in the Params property.
<a href="#">Params</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#">Plan</a>	Used to get or set the PLAN clause of the SELECT statement.
<a href="#">Prepare</a> (inherited from <a href="#">TCustomDADataset</a> )	Allocates, opens, and parses cursor for a query.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">ReadOnly</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#">RefreshOptions</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate when the editing record is refreshed.
<a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )	Actuali es field values for the current record.
<a href="#">RestoreSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Restores the SQL property modified by AddWhere and SetOrderBy.

<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchronizes the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">RowsAffected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#">RowsDeleted</a>	Used to indicate the number of rows that were deleted during the last query operation.
<a href="#">RowsFetched</a>	Used to get the number of the currently fetched rows.
<a href="#">RowsInserted</a>	Used to indicate the number of rows that were inserted during the last query operation.
<a href="#">RowsUpdated</a>	Used to indicate the number of rows that were updated during the last query operation.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to provide a SQL statement that a query component executes when its Open method is called.
<a href="#">SQLDelete</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.
<a href="#">SQLInsert</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.

<a href="#">SQLLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to perform a record lock.
<a href="#">SQLRefresh</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#">TCustomDADataset.RefreshRecord</a> procedure.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">SQLType</a>	Used to get the typecode of the SQL statement being processed by the InterBase database server.
<a href="#">SQLUpdate</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#">Transaction</a>	Used to determine the transaction under which the query of this dataset executes.
<a href="#">UniDirectional</a> (inherited from <a href="#">TCustomDADataset</a> )	Used if an application does not need bidirectional access to records in the result set.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the <a href="#">ApplyUpdates</a> method while cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.
<a href="#">UpdateTransaction</a>	Used to get or set the transaction for modifying a dataset.

## See Also

- [TCustomIBCDataset Class](#)
  - [TCustomIBCDataset Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1.2.1 AutoCommit Property

Used to automatically commit each update, insert or delete statement by database server.

#### Class

[TCustomIBCDataset](#)

#### Syntax

```
property AutoCommit: boolean;
```

#### Remarks

When True and Connection.AutoCommit is True, each update, insert or delete statement is automatically committed by database server.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1.2.2 Connection Property

Used to specify the connection in which the dataset will be executed.

#### Class

[TCustomIBCDataset](#)

#### Syntax

```
property Connection: TIBCCConnection;
```

#### Remarks

Use the Connection property to specify the connection in which the dataset will be executed. If connection is not connected, the Open method calls Connection.Connect.

#### See Also

- [TIBCCConnection](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1.2.3 Cursor Property

Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.

#### Class

[TCustomIBCDataset](#)

## Syntax

**property** Cursor: **string**;

## Remarks

Use the Cursor property for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause. FetchRows property must be set to 1 for using cursor.

## Example

The following example shows how to use the Cursor property. It is used for enumerating table rows. There are TIBCCConnection and TIBCTransaction components in application with already open connection in this example.

```
var
  i : integer;
....
Query.SQL.Text := 'SELECT * FROM MyTable FOR UPDATE';
Query.FetchRows := 1;
Query.Open;
i:=1;
while not Query.EOF do begin
  //process fetched data
  Connection.ExecSQL('UPDATE MyTable SET MyField = ' + IntToStr(i) +
                    ' WHERE CURRENT OF ' + Query.Cursor , []);

  Query.Next;
  i:=i+1;
end;
Connection.DefaultTransaction.Commit;
Query.Close;
```

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1.2.4 DMLRefresh Property

Used to refresh record by the RETURNIG clause when insert is performed.

## Class

[TCustomIBCDDataSet](#)

## Syntax

**property** DMLRefresh: **boolean**;

## Remarks

Use the DMLRefresh property to refresh record by the RETURNIG clause when insert is performed. This feature is only for Firebird 2.0 and higher. The default value is False.

**Note:** When DMLRefresh property is set to True, [TCustomDADDataSet.RefreshOptions](#) should be set to False to avoid rereading fields' values from the server.

## See Also

- [Updating Data with IBDAC Dataset Components](#)
  - [TCustomDADataset.RefreshOptions](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1.2.5 GeneratorMode Property

Used to specify which method is used internally to generate a sequenced field.

## Class

[TCustomIBCDataset](#)

## Syntax

```
property GeneratorMode: TGeneratorMode default gmPost;
```

## Remarks

Set the GeneratorMode property to specify which method is used internally to generate a sequenced field.

## See Also

- [KeyGenerator](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1.2.6 GeneratorStep Property

Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.

## Class

[TCustomIBCDataset](#)

## Syntax

```
property GeneratorStep: integer default 1;
```

## Remarks

Use the GeneratorStep property to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature. The default value is 1.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1.2.7 Handle Property

Used to specify the handle for the SQL statement of TCustomIBCDataset.

## Class



[TCustomIBCDataset](#)

### Syntax

**property** Handle: TISC\_STMT\_HANDLE;

### Remarks

Use the Handle property to specify the handle for the SQL statement of TCustomIBCDataset.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1.2.8 IsQuery Property

Used to check if the SQL statement returns rows.

### Class

[TCustomIBCDataset](#)

### Syntax

**property** IsQuery: boolean;

### Remarks

When the TCustomIBCDataset component is prepared, returns True, if the SQL statement is a SELECT block that returns the REF CURSOR parameter. Use the IsQuery property to check whether the SQL statement returns rows or not. TCustomIBCDataset returns rows when the SQL statement is the SELECT or PL/SQL block with the REF CURSOR parameter. TCustomIBCDataset must be prepared before. IsQuery is a read-only property.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1.2.9 KeyGenerator Property

Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.

### Class

[TCustomIBCDataset](#)

### Syntax

**property** KeyGenerator: string;

### Remarks

Use the KeyGenerator property to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.

**Note:** KeyGenerator is used by TCustomIBCDataset only if [TCustomDADataset.KeyFields](#) property is assigned.

### See Also

- [GeneratorMode](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1.2.10 LockMode Property

Used to indicate when to perform a locking of editing record.

### Class

[TCustomIBCDataset](#)

### Syntax

```
property LockMode: TLockMode default lmNone;
```

### Remarks

Use the LockMode property to define when to perform locking of an editing record. Locking a record is useful in creating multi-user applications. It prevents modification of a record by several users at the same time. Locking realises through execution of the SELECT FOR UPDATE statement in Firebird 1.5 and through the UPDATE operation in other database servers.

### See Also

- [TCustomDADataset.Lock](#)
  - [TCustomDADataset.SQLLock](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1.2.11 Options Property

Used to specify the behaviour of the TCustomIBCDataset object.

### Class

[TCustomIBCDataset](#)

### Syntax

```
property Options: TIBCDatasetOptions;
```

### Remarks

Set the properties of Options to specify behaviour of a TCustomIBCDataset object. Descriptions of all options are in the table below.

Option Name	Description
<a href="#">AutoClose</a>	Used to for CustomIBCDataset to close cursor after fetching all rows.
<a href="#">BooleanDomainFields</a>	Used to create TBooleanField for fields that have domain of the integer data type, and the domain name contains 'BOOLEAN'.

[CacheArrays](#)

Used to allocate local memory buffer to hold a copy of the array content.

[CacheBlobs](#)

Used to allocate local memory buffer to hold a copy of the BLOB content.

[ComplexArrayFields](#)

Used to store array fields as TIBCArraryField objects.

[DefaultValues](#)

Used for TCustomIBCDDataSet to fill the DefaultExpression property of TField objects by the appropriate value.

[DeferredArrayRead](#)

Used for fetching all InterBase array values when they are explicitly requested.

[DeferredBlobRead](#)

Used for fetching all InterBase BLOB values when they are explicitly requested.

[DescribeParams](#)

Used to specify whether to query [TIBCPParam](#) properties from the server when executing the [TCustomDADataset.Prepare](#) method.

[FieldsAsString](#)

Used to treat all non-BLOB fields as being of string datatype.

[FullRefresh](#)

Used to refresh fields from all tables of the query.

[StreamedBlobs](#)

Used to save all edited BLOBs as streamed and to handle the streamed

[StrictUpdate](#)

Used for TCustomIBCDDataSet to raise an exception when the number of the updated or deleted records is not equal 1.

## See Also

- [TCustomDADataset.Options](#)
- [BLOB Data Types](#)

©

1997-2013 Devart. All Rights Reserved.

17.13.1.1.2.12 Plan Property

Used to get or set the PLAN clause of the SELECT statement.

## Class

[TCustomIBCDataset](#)

**Syntax**

```
property Plan: string;
```

**Remarks**

Use the Plan property to get or set the PLAN clause of the SELECT statement.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.1.2.13 RowsDeleted Property

Used to indicate the number of rows that were deleted during the last query operation.

**Class**

[TCustomIBCDataset](#)

**Syntax**

```
property RowsDeleted: integer;
```

**Remarks**

Check RowsDeleted to determine how many rows were deleted during the last query operation. If RowsDeleted is -1, the query has not deleted any rows.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.1.2.14 RowsFetched Property

Used to get the number of the currently fetched rows.

**Class**

[TCustomIBCDataset](#)

**Syntax**

```
property RowsFetched: integer;
```

**Remarks**

Use the RowsFetched property to get the number of the currently fetched rows.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.1.2.15 RowsInserted Property

Used to indicate the number of rows that were inserted during the last query operation.

**Class**

[TCustomIBCDataset](#)

**Syntax**

```
property RowsInserted: integer;
```

---

## Remarks

Check RowsInserted to determine how many rows were inserted during the last query operation. If RowsInserted is -1, the query has not inserted any rows.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1.2.16 RowsUpdated Property

Used to indicate the number of rows that were updated during the last query operation.

## Class

[TCustomIBCDataset](#)

## Syntax

```
property RowsUpdated: integer;
```

## Remarks

Check RowsUpdated to determine how many rows were updated during the last query operation. If RowsUpdated is -1, the query has not updated any rows.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1.2.17 SQLType Property

Used to get the typecode of the SQL statement being processed by the InterBase database server.

## Class

[TCustomIBCDataset](#)

## Syntax

```
property SQLType: integer;
```

## Remarks

Read the SQLType property to get the typecode of the SQL statement being processed by the InterBase database server.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1.2.18 Transaction Property

Used to determine the transaction under which the query of this dataset executes.

## Class

[TCustomIBCDataset](#)

## Syntax

```
property Transaction: TIBCTransaction stored IsTransactionStored;
```

## Remarks

Use the Transaction property to determine the transaction under which the query of this dataset executes. You can separately set transaction for executing modifying queries with the [UpdateTransaction](#) property. By default the Transaction and the UpdateTransaction properties are the same.

## See Also

- [UpdateTransaction](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1.2.19 UpdateTransaction Property

Used to get or set the transaction for modifying a dataset.

## Class

[TCustomIBCDataset](#)

## Syntax

```
property UpdateTransaction: TIBCTransaction;
```

## Remarks

Use the UpdateTransaction property to set or get the transaction for modifying a dataset. By default UpdateTransaction is the same as [Transaction](#).

## See Also

- [Transaction](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1.3 Methods

Methods of the **TCustomIBCDataset** class.

For a complete list of the **TCustomIBCDataset** class members, see the [TCustomIBCDataset Members](#) topic.

## Public

Name	Description
<a href="#">AddWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after dataset finishes fetching data from server.

<a href="#">AfterUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">BaseSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#">BeforeFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#">BeforeUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#">BreakExec</a> (inherited from <a href="#">TCustomDADataset</a> )	Breaks execution of the SQL statement on the server.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">Connection</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a connection object to use to connect to a data store.
<a href="#">CreateBlobStream</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#">CreateProcCall</a>	Assigns PL/SQL block that calls stored procedure to the SQL property.
<a href="#">Debug</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">DeleteWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.

<a href="#">DetailFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#">Disconnected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to keep dataset opened after connection is closed.
<a href="#">Encryption</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the options of the data encryption in a dataset.
<a href="#">Execute</a> (inherited from <a href="#">TCustomDADataset</a> )	Executes a SQL statement on the server.
<a href="#">Executing</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether SQL statement is still being executed.
<a href="#">Fetched</a>	Used to determine if TCustomDADataset has already fetched all rows.
<a href="#">Fetching</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#">FetchingAll</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#">FetchRows</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to define the number of rows to be transferred across the network at the same time.
<a href="#">FilterSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#">FinalSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#">FindKey</a> (inherited from <a href="#">TCustomDADataset</a> )	Searches for a record which contains specified field values.
<a href="#">FindMacro</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether a specified macro exists in a dataset.
<a href="#">FindNearest</a> (inherited from <a href="#">TCustomDADataset</a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.



<a href="#">FindParam</a>	Determines if a parameter with the specified name exists in a dataset.
<a href="#">GetArray</a>	Retrieves a TIBCArry object for a field when only its name is known.
<a href="#">GetBlob</a>	Retrieves a TIBCBlob object for a field when only its name is known.
<a href="#">GetDataType</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#">GetFieldObject</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns a multireference shared object from field.
<a href="#">GetFieldPrecision</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the precision of a number field.
<a href="#">GetFieldScale</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the scale of a number field.
<a href="#">GetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves an ORDER BY clause from a SQL statement.
<a href="#">GotoCurrent</a> (inherited from <a href="#">TCustomDADataset</a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">IsQuery</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to check whether SQL statement returns rows.
<a href="#">KeyFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.

<a href="#"><u>LocateEx</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Excludes features that don't need to be included to the <a href="#"><u>TMemDataSet.Locate</u></a> method of TDataSet.
<a href="#"><u>Lock</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Locks the current record.
<a href="#"><u>MacroByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Finds a Macro with the name passed in Name.
<a href="#"><u>MacroCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to get the number of macros associated with the Macros property.
<a href="#"><u>Macros</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Makes it possible to change SQL queries easily.
<a href="#"><u>MasterFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#"><u>MasterSource</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the data source component which binds current dataset to the master one.
<a href="#"><u>OnUpdateError</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#"><u>OnUpdateRecord</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when a single update component can not handle the updates.
<a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the behaviour of TCustomDADataset object.
<a href="#"><u>ParamByName</u></a>	Called to set or use parameter information for a specific parameter based on its name.
<a href="#"><u>ParamCheck</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#"><u>ParamCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate how many parameters are there in the Params property.
<a href="#"><u>Params</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to view and set parameter names, values, and data types dynamically.

<a href="#">Prepare</a> (inherited from <a href="#">TCustomDADataset</a> )	Allocates, opens, and parses cursor for a query.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">ReadOnly</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#">RefreshOptions</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate when the editing record is refreshed.
<a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )	Actuali es field values for the current record.
<a href="#">RestoreSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">RowsAffected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to provide a SQL statement that a query component executes when its Open method is called.
<a href="#">SQLDelete</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.

<a href="#">SQLInsert</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#">SQLLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to perform a record lock.
<a href="#">SQLRefresh</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#">TCustomDADataset.RefreshRecord</a> procedure.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">SQLUpdate</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#">UniDirectional</a> (inherited from <a href="#">TCustomDADataset</a> )	Used if an application does not need bidirectional access to records in the result set.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the <a href="#">ApplyUpdates</a> method while cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.

### See Also

- [TCustomIBCDataset Class](#)
- [TCustomIBCDataset Class Members](#)

## 17.13.1.1.3.1 CreateProcCall Method

Assigns PL/SQL block that calls stored procedure to the SQL property.

**Class**

[TCustomIBCDataset](#)

**Syntax**

```
procedure CreateProcCall(const Name: string);
```

**Parameters**

*Name*

Holds the name of the stored procedure

**Remarks**

Call the CreateProcCall method to assign PL/SQL block that calls stored procedure specified by Name to the SQL property. Retrieves the information about parameters of the procedure from InterBase. After calling CreateProcCall you can execute stored procedure by Execute method.

**See Also**

- [TCustomDADataset.Execute](#)
- [TCustomDAConnection.ExecProc](#)
- [TIBCStoredProc](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.1.3.2 Fetched Method

Used to determine if TCustomDADataset has already fetched all rows.

**Class**

[TCustomIBCDataset](#)

**Syntax**

```
function Fetched: boolean; override;
```

**Return Value**

True, if TCustomDADataset has already fetched all rows.

**Remarks**

Call the Fetched method to learn whether TCustomDADataset has already fetched all rows or not.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.1.3.3 FindParam Method

Determines if a parameter with the specified name exists in a dataset.

**Class**

[TCustomIBCDataset](#)

### Syntax

```
function FindParam(const Value: string): TIBCPParam;
```

#### Parameters

*Value*

Holds the name of the param for which to search.

#### Return Value

The TIBCPParam object for the specified Name, if a param with a matching name was found. Nil otherwise.

### Remarks

Call the FindParam method to determine if a parameter with the specified name exists in a dataset. Name is the name of the param for which to search. If FindParam finds a param with a matching name, it returns the TIBCPParam object for the specified Name. Otherwise it returns nil.

### See Also

- [TCustomDADataset.Params](#)
  - [ParamByName](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1.3.4 GetArray Method

Retrieves a TIBCArray object for a field when only its name is known.

### Class

[TCustomIBCDataset](#)

### Syntax

```
function GetArray(FieldDesc: TFieldDesc): TIBCArray; overload;  
function GetArray(const FieldName: string): TIBCArray; overload;
```

#### Parameters

*FieldName*

Holds the field name of an existing field.

#### Return Value

The TIBCArray object, if a field with a matching name was found.

### Remarks

Call the GetArray method to retrieve a TIBCArray object for a field when only its name is known. FieldName is the name of an existing field. The field should have the ftIBCArray type.

## See Also

- [TIBCArray](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1.3.5 GetBlob Method

Retrieves a TIBCBlob object for a field when only its name is known.

## Class

[TCustomIBCDataset](#)

## Syntax

```
function GetBlob(const FieldName: string): TIBCBlob;
```

### Parameters

*FieldName*

Holds the field name of an existing field.

### Return Value

The TIBCBlob object, if a field with a matching name was found.

## Remarks

Call the GetBlob method to retrieve a TIBCBlob object for a field when only its name is known. FieldName is the name of an existing field. The field should have the ftIBCBlob type.

## See Also

- [TIBCBlob](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1.3.6 ParamByName Method

Called to set or use parameter information for a specific parameter based on its name.

## Class

[TCustomIBCDataset](#)

## Syntax

```
function ParamByName(const Value: string): TIBCParm;
```

### Parameters

*Value*

Holds the name of the parameter for which to retrieve information.

### Return Value

A object, if there is a parameter with a matching name.

### Remarks

Call the ParamByName method to set or use parameter information for a specific parameter based on its name. Name is the name of the parameter for which to retrieve information. ParamByName is used to set a parameter's value at runtime and returns a [TIBCPParam](#) object.

### Example

For example, the following statement retrieves the current value of a parameter called "Contact" into an edit box:

```
Edit1.Text := Query1.ParamsByName('Contact').AsString;
```

### See Also

- [TIBCPParam](#)
  - [TCustomDADataset.Params](#)
  - [TCustomDADataset.FindParam](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.2 TCustomIBCQuery Class

A base class for defining functionality for descendant classes which access database using SQL statements. c

For a list of all members of this type, see [TCustomIBCQuery](#) members.

### Unit

[IBC](#)

### Syntax

```
TCustomIBCQuery = class (TCustomIBCDataSet);
```

### Remarks

TCustomIBCQuery is a base class that defines functionality for descendant classes which access database using SQL statements. Applications never use the TCustomIBCQuery objects directly. Instead they use descendants of TCustomIBCQuery, such as TIBCQuery, TIBCStoredProc and TIBCTable. TCustomIBCQuery implements functionality to update database tables using DML SQL statements. Put SQL statements into SQLInsert, SQLDelete, SQLUpdate properties. There is no restriction on their syntax, so any SQL statements are allowed. Usually you need to use INSERT, DELETE and UPDATE statements but also you can use stored procedures in more diverse cases. SQLInsert, SQLDelete, SQLUpdate, SQLLock, SQLRefresh properties support automatic binding of parameters which have names identical to fields captions. To retrieve the value of a field as it was before operation use the field name with 'OLD '. This is especially useful when doing field comparisons in the WHERE clause



of the statement. Use [TCustomDADataset.BeforeUpdateExecute](#) event to assign value to additional parameters and [TCustomDADataset.AfterUpdateExecute](#) event for reading them.

TCustomIBCQuery is read-only when none of SQLInsert, SQLDelete, SQLUpdate properties are defined.

## Inheritance Hierarchy

[TMemDataSet](#)  
[TCustomDADataset](#)  
[TCustomIBCDataset](#)  
**TCustomIBCQuery**

## See Also

- [TCustomIBCDataset](#)
- [TIBCQuery](#)
- [TIBCStoredProc](#)
- [TIBCTable](#)

© 1997-2013 Devart. All Rights Reserved.

17.13.1.2.1 Members

[TCustomIBCQuery](#) class overview.

## Properties

Name	Description
<a href="#">AutoCommit</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to automatically commit each update, insert or delete statement by database server.
<a href="#">BaseSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">Connection</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the connection in which the dataset will be executed.
<a href="#">Cursor</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.

<a href="#">Debug</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#">DetailFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#">Disconnected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to keep dataset opened after connection is closed.
<a href="#">DMLRefresh</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to refresh record by the RETURNIG clause when insert is performed.
<a href="#">Encryption</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the options of the data encryption in a dataset.
<a href="#">FetchRows</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to define the number of rows to be transferred across the network at the same time.
<a href="#">FilterSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#">FinalSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#">GeneratorMode</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify which method is used internally to generate a sequenced field.
<a href="#">GeneratorStep</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">IsQuery</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to check if the SQL statement returns rows.

<a href="#">KeyFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#">KeyGenerator</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">LockMode</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate when to perform a locking of editing record.
<a href="#">MacroCount</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to get the number of macros associated with the Macros property.
<a href="#">Macros</a> (inherited from <a href="#">TCustomDADataset</a> )	Makes it possible to change SQL queries easily.
<a href="#">MasterFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#">MasterSource</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the data source component which binds current dataset to the master one.
<a href="#">Options</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#">ParamCheck</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#">ParamCount</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate how many parameters are there in the Params property.

<a href="#">Params</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#">Plan</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get or set the PLAN clause of the SELECT statement.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">ReadOnly</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#">RefreshOptions</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate when the editing record is refreshed.
<a href="#">RowsAffected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#">RowsDeleted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were deleted during the last query operation.
<a href="#">RowsFetched</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the number of the currently fetched rows.
<a href="#">RowsInserted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were inserted during the last query operation.
<a href="#">RowsUpdated</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were updated during the last query operation.
<a href="#">SQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to provide a SQL statement that a query component executes when its Open method is called.
<a href="#">SQLDelete</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.
<a href="#">SQLInsert</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#">SQLLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to perform a record lock.

<a href="#">SQLRefresh</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#">TCustomDADataset.RefreshRecord</a> procedure.
<a href="#">SQLType</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the typecode of the SQL statement being processed by the InterBase database server.
<a href="#">SQLUpdate</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#">Transaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine the transaction under which the query of this dataset executes.
<a href="#">UniDirectional</a> (inherited from <a href="#">TCustomDADataset</a> )	Used if an application does not need bidirectional access to records in the result set.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataset</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataset</a> )	Used to check the status of the cached updates buffer.
<a href="#">UpdateTransaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get or set the transaction for modifying a dataset.

## Methods

Name	Description
<a href="#">AddWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataset</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">BreakExec</a> (inherited from <a href="#">TCustomDADataset</a> )	Breaks execution of the SQL statement on the server.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataset</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataset</a> )	Clears the cached updates buffer.

<a href="#"><u>CreateBlobStream</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#"><u>CreateProcCall</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Assigns PL/SQL block that calls stored procedure to the SQL property.
<a href="#"><u>DeferredPost</u></a> (inherited from <a href="#"><u>TMemDataset</u></a> )	Makes permanent changes to the database server.
<a href="#"><u>DeleteWhere</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#"><u>Execute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Executes a SQL statement on the server.
<a href="#"><u>Executing</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether SQL statement is still being executed.
<a href="#"><u>Fetches</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to determine if TCustomDADataset has already fetched all rows.
<a href="#"><u>Fetching</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#"><u>FetchingAll</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#"><u>FindKey</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Searches for a record which contains specified field values.
<a href="#"><u>FindMacro</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether a specified macro exists in a dataset.
<a href="#"><u>FindNearest</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#"><u>FindParam</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Determines if a parameter with the specified name exists in a dataset.
<a href="#"><u>GetArray</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Retrieves a TIBCArrary object for a field when only its name is known.
<a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Retrieves a TIBCBlob object for a field when only its name is known.

<a href="#">GetDataType</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#">GetFieldObject</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns a multireference shared object from field.
<a href="#">GetFieldPrecision</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the precision of a number field.
<a href="#">GetFieldScale</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the scale of a number field.
<a href="#">GetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves an ORDER BY clause from a SQL statement.
<a href="#">GotoCurrent</a> (inherited from <a href="#">TCustomDADataset</a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">Lock</a> (inherited from <a href="#">TCustomDADataset</a> )	Locks the current record.
<a href="#">MacroByName</a> (inherited from <a href="#">TCustomDADataset</a> )	Finds a Macro with the name passed in Name.
<a href="#">ParamByName</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Called to set or use parameter information for a specific parameter based on its name.
<a href="#">Prepare</a> (inherited from <a href="#">TCustomDADataset</a> )	Allocates, opens, and parses cursor for a query.
<a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )	Actuali es field values for the current record.
<a href="#">RestoreSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.

<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.

## Events

Name	Description
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">BeforeFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#">BeforeUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.



[OnUpdateRecord](#) (inherited from [TMemDataSet](#)) Occurs when a single update component can not handle the updates.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.3 TCustomIBCTable Class

A base class that defines functionality for descendant classes which access data in a single table without writing SQL statements.

For a list of all members of this type, see [TCustomIBCTable](#) members.

#### Unit

[IBC](#)

#### Syntax

```
TCustomIBCTable = class(TCustomIBCQuery);
```

#### Remarks

TCustomIBCTable component is inherited from TCustomIBCQuery class.

Applications never use TCustomIBCTable objects directly. Instead they use descendants of TCustomIBCTable such as TIBCTable.

It allows to retrieve and update data in single table without writing SQL statements.

Use TableName to specify the name of a table. TIBCTable uses KeyFields property to build SQL statements for updating table data. KeyFields is a string containing a semicolon-delimited list of field names. If KeyFields is not defined before opening, TIBCTable uses primary or unique key.

#### Inheritance Hierarchy

```

TMemDataSet
  TCustomDADataset
    TCustomIBCDataset
      TCustomIBCQuery
        TCustomIBCTable

```

#### See Also

- [TIBCTable](#)
- [TCustomIBCDataset](#)
- [TIBCQuery](#)
- [Master/Detail Relationships](#)
- [Updating Data with IBDAc Dataset Components](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.3.1 Members

[TCustomIBCTable](#) class overview.

#### Properties

Name	Description
<a href="#"><u>AutoCommit</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to automatically commit each update, insert or delete statement by database server.
<a href="#"><u>BaseSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#"><u>CachedUpdates</u></a> (inherited from <a href="#"><u>TMemDataset</u></a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#"><u>Connection</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the connection in which the dataset will be executed.
<a href="#"><u>Cursor</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
<a href="#"><u>Debug</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#"><u>DetailFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#"><u>Disconnected</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to keep dataset opened after connection is closed.
<a href="#"><u>DMLRefresh</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to refresh record by the RETURNIG clause when insert is performed.
<a href="#"><u>Encryption</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the options of the data encryption in a dataset.
<a href="#"><u>Exists</u></a>	Indicates whether a table with the name passed in TableName exists in the database.
<a href="#"><u>FetchRows</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to define the number of rows to be transferred across the network at the same time.
<a href="#"><u>FilterSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to change the WHERE clause of SELECT statement and reopen a query.

<a href="#"><u>FinalSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#"><u>GeneratorMode</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify which method is used internally to generate a sequenced field.
<a href="#"><u>GeneratorStep</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
<a href="#"><u>Handle</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#"><u>IndexFieldNames</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to check if the SQL statement returns rows.
<a href="#"><u>KeyFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#"><u>KeyGenerator</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#"><u>LocalConstraints</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#"><u>LocalUpdate</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to prevent implicit update of rows on database server.
<a href="#"><u>LockMode</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to indicate when to perform a locking of editing record.
<a href="#"><u>MacroCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to get the number of macros associated with the Macros property.
<a href="#"><u>Macros</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Makes it possible to change SQL queries easily.

<a href="#"><u>MasterFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#"><u>MasterSource</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the data source component which binds current dataset to the master one.
<a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#"><u>ParamCheck</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#"><u>ParamCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate how many parameters are there in the Params property.
<a href="#"><u>Params</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#"><u>Plan</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get or set the PLAN clause of the SELECT statement.
<a href="#"><u>Prepared</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Determines whether a query is prepared for execution or not.
<a href="#"><u>ReadOnly</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#"><u>RefreshOptions</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate when the editing record is refreshed.
<a href="#"><u>RowsAffected</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#"><u>RowsDeleted</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to indicate the number of rows that were deleted during the last query operation.
<a href="#"><u>RowsFetched</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get the number of the currently fetched rows.
<a href="#"><u>RowsInserted</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to indicate the number of rows that were inserted during the last query operation.

<a href="#"><u>RowsUpdated</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to indicate the number of rows that were updated during the last query operation.
<a href="#"><u>SQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to provide a SQL statement that a query component executes when its Open method is called.
<a href="#"><u>SQLDelete</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.
<a href="#"><u>SQLInsert</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#"><u>SQLLock</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify a SQL statement that will be used to perform a record lock.
<a href="#"><u>SQLRefresh</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#"><u>TCustomDADataset.RefreshRecord</u></a> procedure.
<a href="#"><u>SQLType</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get the typecode of the SQL statement being processed by the InterBase database server.
<a href="#"><u>SQLUpdate</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#"><u>Transaction</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to determine the transaction under which the query of this dataset executes.
<a href="#"><u>UniDirectional</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used if an application does not need bidirectional access to records in the result set.
<a href="#"><u>UpdateRecordTypes</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#"><u>UpdatesPending</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to check the status of the cached updates buffer.
<a href="#"><u>UpdateTransaction</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get or set the transaction for modifying a dataset.

## Methods

Name	Description
<a href="#"><u>AddWhere</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#"><u>ApplyUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#"><u>BreakExec</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Breaks execution of the SQL statement on the server.
<a href="#"><u>CancelUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#"><u>CommitUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Clears the cached updates buffer.
<a href="#"><u>CreateBlobStream</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#"><u>CreateProcCall</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Assigns PL/SQL block that calls stored procedure to the SQL property.
<a href="#"><u>DeferredPost</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Makes permanent changes to the database server.
<a href="#"><u>DeleteTable</u></a>	Deletes a table from a database.
<a href="#"><u>DeleteWhere</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#"><u>EmptyTable</u></a>	Truncates the current table content on the server.
<a href="#"><u>Execute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Executes a SQL statement on the server.
<a href="#"><u>Executing</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether SQL statement is still being executed.
<a href="#"><u>Fetches</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to determine if TCustomDADataset has already fetched all rows.
<a href="#"><u>Fetching</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is still fetching rows.

<a href="#">FetchingAll</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#">FindKey</a> (inherited from <a href="#">TCustomDADataset</a> )	Searches for a record which contains specified field values.
<a href="#">FindMacro</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether a specified macro exists in a dataset.
<a href="#">FindNearest</a> (inherited from <a href="#">TCustomDADataset</a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#">FindParam</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Determines if a parameter with the specified name exists in a dataset.
<a href="#">GetArray</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Retrieves a TIBCArrary object for a field when only its name is known.
<a href="#">GetBlob</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Retrieves a TIBCBlob object for a field when only its name is known.
<a href="#">GetDataType</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#">GetFieldObject</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns a multireference shared object from field.
<a href="#">GetFieldPrecision</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the precision of a number field.
<a href="#">GetFieldScale</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the scale of a number field.
<a href="#">GetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves an ORDER BY clause from a SQL statement.
<a href="#">GotoCurrent</a> (inherited from <a href="#">TCustomDADataset</a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.

<a href="#">Lock</a> (inherited from <a href="#">TCustomDADataset</a> )	Locks the current record.
<a href="#">MacroByName</a> (inherited from <a href="#">TCustomDADataset</a> )	Finds a Macro with the name passed in Name.
<a href="#">ParamByName</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Called to set or use parameter information for a specific parameter based on its name.
<a href="#">Prepare</a> (inherited from <a href="#">TCustomDADataset</a> )	Allocates, opens, and parses cursor for a query.
<a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )	Actuali es field values for the current record.
<a href="#">RestoreSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.



## Events

Name	Description
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">BeforeFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#">BeforeUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.3.2 Properties

Properties of the **TCustomIBCTable** class.

For a complete list of the **TCustomIBCTable** class members, see the [TCustomIBCTable Members](#) topic.

## Public

Name	Description
<a href="#">AddWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.

<a href="#"><u>AutoCommit</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to automatically commit each update, insert or delete statement by database server.
<a href="#"><u>BaseSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#"><u>BeforeFetch</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#"><u>BeforeUpdateExecute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#"><u>BreakExec</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Breaks execution of the SQL statement on the server.
<a href="#"><u>CachedUpdates</u></a> (inherited from <a href="#"><u>TMemDataset</u></a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#"><u>CancelUpdates</u></a> (inherited from <a href="#"><u>TMemDataset</u></a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#"><u>CommitUpdates</u></a> (inherited from <a href="#"><u>TMemDataset</u></a> )	Clears the cached updates buffer.
<a href="#"><u>Connection</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the connection in which the dataset will be executed.
<a href="#"><u>CreateBlobStream</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#"><u>CreateProcCall</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Assigns PL/SQL block that calls stored procedure to the SQL property.
<a href="#"><u>Cursor</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
<a href="#"><u>Debug</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#"><u>DeferredPost</u></a> (inherited from <a href="#"><u>TMemDataset</u></a> )	Makes permanent changes to the database server.

<a href="#">DeleteWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#">DetailFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#">Disconnected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to keep dataset opened after connection is closed.
<a href="#">DMLRefresh</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to refresh record by the RETURNIG clause when insert is performed.
<a href="#">Encryption</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the options of the data encryption in a dataset.
<a href="#">Execute</a> (inherited from <a href="#">TCustomDADataset</a> )	Executes a SQL statement on the server.
<a href="#">Executing</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether SQL statement is still being executed.
<a href="#">Exists</a>	Indicates whether a table with the name passed in TableName exists in the database.
<a href="#">Fetched</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine if TCustomDADataset has already fetched all rows.
<a href="#">Fetching</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#">FetchingAll</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#">FetchRows</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to define the number of rows to be transferred across the network at the same time.
<a href="#">FilterSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#">FinalSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#">FindKey</a> (inherited from <a href="#">TCustomDADataset</a> )	Searches for a record which contains specified field values.

<a href="#">FindMacro</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether a specified macro exists in a dataset.
<a href="#">FindNearest</a> (inherited from <a href="#">TCustomDADataset</a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#">FindParam</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Determines if a parameter with the specified name exists in a dataset.
<a href="#">GeneratorMode</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify which method is used internally to generate a sequenced field.
<a href="#">GeneratorStep</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
<a href="#">GetArray</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Retrieves a TIBCArrary object for a field when only its name is known.
<a href="#">GetBlob</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Retrieves a TIBCBlob object for a field when only its name is known.
<a href="#">GetDataTypes</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#">GetFieldObject</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns a multireference shared object from field.
<a href="#">GetFieldPrecision</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the precision of a number field.
<a href="#">GetFieldScale</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the scale of a number field.
<a href="#">GetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves an ORDER BY clause from a SQL statement.
<a href="#">GotoCurrent</a> (inherited from <a href="#">TCustomDADataset</a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.

<a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to check if the SQL statement returns rows.
<a href="#"><u>KeyFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#"><u>KeyGenerator</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#"><u>LocalConstraints</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#"><u>LocalUpdate</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to prevent implicit update of rows on database server.
<a href="#"><u>Locate</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#"><u>LocateEx</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Excludes features that don't need to be included to the <a href="#"><u>TMemDataSet.Locate</u></a> method of TDataSet.
<a href="#"><u>Lock</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Locks the current record.
<a href="#"><u>LockMode</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to indicate when to perform a locking of editing record.
<a href="#"><u>MacroByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Finds a Macro with the name passed in Name.
<a href="#"><u>MacroCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to get the number of macros associated with the Macros property.
<a href="#"><u>Macros</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Makes it possible to change SQL queries easily.
<a href="#"><u>MasterFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.

<a href="#"><u>MasterSource</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the data source component which binds current dataset to the master one.
<a href="#"><u>OnUpdateError</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#"><u>OnUpdateRecord</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when a single update component can not handle the updates.
<a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Called to set or use parameter information for a specific parameter based on its name.
<a href="#"><u>ParamCheck</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#"><u>ParamCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate how many parameters are there in the Params property.
<a href="#"><u>Params</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#"><u>Plan</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get or set the PLAN clause of the SELECT statement.
<a href="#"><u>Prepare</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Allocates, opens, and parses cursor for a query.
<a href="#"><u>Prepared</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Determines whether a query is prepared for execution or not.
<a href="#"><u>ReadOnly</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#"><u>RefreshOptions</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate when the editing record is refreshed.
<a href="#"><u>RefreshRecord</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Actuali es field values for the current record.
<a href="#"><u>RestoreSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#"><u>RestoreUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Marks all records in the cache of updates as unapplied.

<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchronizes the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">RowsAffected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#">RowsDeleted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were deleted during the last query operation.
<a href="#">RowsFetched</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the number of the currently fetched rows.
<a href="#">RowsInserted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were inserted during the last query operation.
<a href="#">RowsUpdated</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were updated during the last query operation.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to provide a SQL statement that a query component executes when its Open method is called.
<a href="#">SQLDelete</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.
<a href="#">SQLInsert</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#">SQLLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to perform a record lock.

<a href="#">SQLRefresh</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#">TCustomDADataset.RefreshRecord</a> procedure.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">SQLType</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the typecode of the SQL statement being processed by the InterBase database server.
<a href="#">SQLUpdate</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#">Transaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine the transaction under which the query of this dataset executes.
<a href="#">UniDirectional</a> (inherited from <a href="#">TCustomDADataset</a> )	Used if an application does not need bidirectional access to records in the result set.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the <a href="#">ApplyUpdates</a> method while cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.
<a href="#">UpdateTransaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get or set the transaction for modifying a dataset.

### See Also

- [TCustomIBCTable Class](#)
- [TCustomIBCTable Class Members](#)



© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.3.2.1 Exists Property

Indicates whether a table with the name passed in TableName exists in the database.

### Class

[TCustomIBCTable](#)

### Syntax

**property** Exists: Boolean;

### Remarks

Use the Exists property to determine whether a table with the name passed in TableName exists in the database.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.3.3 Methods

Methods of the **TCustomIBCTable** class.

For a complete list of the **TCustomIBCTable** class members, see the [TCustomIBCTable Members](#) topic.

### Public

Name	Description
<a href="#">AddWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">AutoCommit</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to automatically commit each update, insert or delete statement by database server.
<a href="#">BaseSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.

<a href="#"><u>BeforeFetch</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#"><u>BeforeUpdateExecute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#"><u>BreakExec</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Breaks execution of the SQL statement on the server.
<a href="#"><u>CachedUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#"><u>CancelUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#"><u>CommitUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Clears the cached updates buffer.
<a href="#"><u>Connection</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the connection in which the dataset will be executed.
<a href="#"><u>CreateBlobStream</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#"><u>CreateProcCall</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Assigns PL/SQL block that calls stored procedure to the SQL property.
<a href="#"><u>Cursor</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
<a href="#"><u>Debug</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#"><u>DeferredPost</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Makes permanent changes to the database server.
<a href="#"><u>DeleteTable</u></a>	Deletes a table from a database.
<a href="#"><u>DeleteWhere</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.

<a href="#">DetailFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#">Disconnected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to keep dataset opened after connection is closed.
<a href="#">DMLRefresh</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to refresh record by the RETURNIG clause when insert is performed.
<a href="#">EmptyTable</a>	Truncates the current table content on the server.
<a href="#">Encryption</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the options of the data encryption in a dataset.
<a href="#">Execute</a> (inherited from <a href="#">TCustomDADataset</a> )	Executes a SQL statement on the server.
<a href="#">Executing</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether SQL statement is still being executed.
<a href="#">Fetched</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine if TCustomDADataset has already fetched all rows.
<a href="#">Fetching</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#">FetchingAll</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#">FetchRows</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to define the number of rows to be transferred across the network at the same time.
<a href="#">FilterSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#">FinalSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#">FindKey</a> (inherited from <a href="#">TCustomDADataset</a> )	Searches for a record which contains specified field values.
<a href="#">FindMacro</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether a specified macro exists in a dataset.

<a href="#"><u>FindNearest</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#"><u>FindParam</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Determines if a parameter with the specified name exists in a dataset.
<a href="#"><u>GeneratorMode</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify which method is used internally to generate a sequenced field.
<a href="#"><u>GeneratorStep</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
<a href="#"><u>GetArray</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Retrieves a TIBCArrary object for a field when only its name is known.
<a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Retrieves a TIBCBlob object for a field when only its name is known.
<a href="#"><u>GetDataType</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#"><u>GetFieldObject</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Returns a multireference shared object from field.
<a href="#"><u>GetFieldPrecision</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves the precision of a number field.
<a href="#"><u>GetFieldScale</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves the scale of a number field.
<a href="#"><u>GetOrderBy</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves an ORDER BY clause from a SQL statement.
<a href="#"><u>GotoCurrent</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#"><u>Handle</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#"><u>IndexFieldNames</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to check if the SQL statement returns rows.

<a href="#"><u>KeyFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#"><u>KeyGenerator</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#"><u>LocalConstraints</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#"><u>LocalUpdate</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to prevent implicit update of rows on database server.
<a href="#"><u>Locate</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#"><u>LocateEx</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Excludes features that don't need to be included to the <a href="#"><u>TMemDataSet.Locate</u></a> method of TDataSet.
<a href="#"><u>Lock</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Locks the current record.
<a href="#"><u>LockMode</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to indicate when to perform a locking of editing record.
<a href="#"><u>MacroByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Finds a Macro with the name passed in Name.
<a href="#"><u>MacroCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to get the number of macros associated with the Macros property.
<a href="#"><u>Macros</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Makes it possible to change SQL queries easily.
<a href="#"><u>MasterFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#"><u>MasterSource</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the data source component which binds current dataset to the master one.

<a href="#"><u>OnUpdateError</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#"><u>OnUpdateRecord</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when a single update component can not handle the updates.
<a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Called to set or use parameter information for a specific parameter based on its name.
<a href="#"><u>ParamCheck</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#"><u>ParamCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate how many parameters are there in the Params property.
<a href="#"><u>Params</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#"><u>Plan</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get or set the PLAN clause of the SELECT statement.
<a href="#"><u>Prepare</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Allocates, opens, and parses cursor for a query.
<a href="#"><u>Prepared</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Determines whether a query is prepared for execution or not.
<a href="#"><u>ReadOnly</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#"><u>RefreshOptions</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate when the editing record is refreshed.
<a href="#"><u>RefreshRecord</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Actuali es field values for the current record.
<a href="#"><u>RestoreSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#"><u>RestoreUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Marks all records in the cache of updates as unapplied.

<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchronizes the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataset</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">RowsAffected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#">RowsDeleted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were deleted during the last query operation.
<a href="#">RowsFetched</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the number of the currently fetched rows.
<a href="#">RowsInserted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were inserted during the last query operation.
<a href="#">RowsUpdated</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were updated during the last query operation.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataset</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to provide a SQL statement that a query component executes when its Open method is called.
<a href="#">SQLDelete</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.
<a href="#">SQLInsert</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#">SQLLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to perform a record lock.

<a href="#">SQLRefresh</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#">TCustomDADataset.RefreshRecord</a> procedure.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">SQLType</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the typecode of the SQL statement being processed by the InterBase database server.
<a href="#">SQLUpdate</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#">Transaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine the transaction under which the query of this dataset executes.
<a href="#">UniDirectional</a> (inherited from <a href="#">TCustomDADataset</a> )	Used if an application does not need bidirectional access to records in the result set.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataset</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataset</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataset</a> )	Reads the status of the latest call to the <a href="#">ApplyUpdates</a> method while cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataset</a> )	Used to check the status of the cached updates buffer.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataset</a> )	Indicates the current update status for the dataset when cached updates are enabled.
<a href="#">UpdateTransaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get or set the transaction for modifying a dataset.

### See Also

- [TCustomIBCTable Class](#)
- [TCustomIBCTable Class Members](#)



© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.3.3.1 DeleteTable Method

Deletes a table from a database.

##### Class

[TCustomIBCTable](#)

##### Syntax

```
procedure DeleteTable;
```

##### Remarks

Call the DeleteTable method to delete a table from database.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.3.3.2 EmptyTable Method

Truncates the current table content on the server.

##### Class

[TCustomIBCTable](#)

##### Syntax

```
procedure EmptyTable;
```

##### Remarks

Call the EmptyTable method to truncate the current table content on the server.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.4 TIBCArray Class

A class representing the value of the InterBase array data type.  
For a list of all members of this type, see [TIBCArray](#) members.

##### Unit

[IBC](#)

##### Syntax

```
TIBCArray = class(TCustomIBCArray);
```

##### Remarks

TIBCArray represents the value of the InterBase array data type. You can get a TIBCArray object by the TCustomIBCDataset.GetArray method after fetching rows contained in an array field. You should explicitly add the [IBCArray](#) unit to 'uses' list to use the TIBCArray class.

##### Inheritance Hierarchy

[TSharedObject](#)

[TDBObject](#)  
[TCustomIBCArrary](#)  
**TIBCArrary**

## See Also

- [TCustomIBCDataset.GetArray](#)
- [TIBParam.AsArray](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.4.1 Members

[TIBCArrary](#) class overview.

## Properties

Name	Description
<a href="#">ArrayDimensions</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Contains the array dimensions count.
<a href="#">ArrayHighBound</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to get or set the upper boundary of the defined dimension subscript.
<a href="#">ArrayID</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Contains the array ID.
<a href="#">ArrayLowBound</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to get or set the lower boundary of the defined dimension subscript.
<a href="#">ArraySize</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to determine the size of the whole array data in bytes.
<a href="#">AsString</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to return array as string.
<a href="#">Cached</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Indicates whether to cache array data on the client side.
<a href="#">CachedDimensions</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Contains cached array dimensions count.
<a href="#">CachedHighBound</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to get the upper boundary of defined dimension subscript of cached array elements.
<a href="#">CachedLowBound</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to get the lower boundary of the defined dimension subscript of cached array elements.
<a href="#">CachedSize</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to get the cached array data size in bytes.
<a href="#">ColumnName</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to get or set the name of the table column that has array type.

<a href="#">DbHandle</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Contains the handle of a database where the array is stored.
<a href="#">IsNull</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to define whether the array field in the database is null.
<a href="#">Items</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to get or set array items.
<a href="#">ItemScale</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to get or set the scale for array items for the NUMERIC and DECIMAL datatypes.
<a href="#">ItemSize</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Contains the size of an array item.
<a href="#">ItemType</a>	Used to return the type of an array element.
<a href="#">Modified</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to indicate if the modifications done in cache were saved to the database.
<a href="#">RefCount</a> (inherited from <a href="#">TSharedObject</a> )	Used to return the count of reference to a TSharedObject object.
<a href="#">TableName</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to set the name of the table containing an array field.
<a href="#">TrHandle</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Contains the handle of the transaction in which the array is read or written.

## Methods

Name	Description
<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">Assign</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Copies the Source object content to the current one.
<a href="#">ClearArray</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Clears all array values on the server if Cached is set to False.
<a href="#">CreateTemporaryArray</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Creates a temporary array on the InterBase server.
<a href="#">GetArrayInfo</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to get array descriptor.
<a href="#">GetItemAsDateTime</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Reads the value of an array item into an object or variable of the TDateTime type.

<a href="#"><u>GetItemAsFloat</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Reads the value of an array item into a floating-point number.
<a href="#"><u>GetItemAsInteger</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Reads the value of an array item into a integer.
<a href="#"><u>GetItemAsSmallInt</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Reads the value of an array item into a short integer.
<a href="#"><u>GetItemAsString</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Reads the value of an array item into a string.
<a href="#"><u>GetItemAsWideString</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Reads the value of an array item into a WideString.
<a href="#"><u>GetItemsSlice</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Returns the array slice items' values.
<a href="#"><u>GetValue</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Returns the array item value.
<a href="#"><u>ReadArray</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Reads an array from the database to memory.
<a href="#"><u>ReadArrayItem</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Reads the array item specified by indices from the database to memory.
<a href="#"><u>ReadArraySlice</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Reads array slice from the database to memory.
<a href="#"><u>Release</u></a> (inherited from <a href="#"><u>TSharedObject</u></a> )	Decrements the reference count.
<a href="#"><u>SetItemAsDateTime</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Assigns the TDateTime value to the contents of an array item.
<a href="#"><u>SetItemAsFloat</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Assigns floating-point value to the contents of an array item.
<a href="#"><u>SetItemAsInteger</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Assigns integer value to the contents of an array item.
<a href="#"><u>SetItemAsSmallInt</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Assigns short integer value to the contents of an array item.
<a href="#"><u>SetItemAsString</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Assigns string value to the contents of an array item.
<a href="#"><u>SetItemAsWideString</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Assigns WideString value to the contents of an array item.
<a href="#"><u>SetItemsSlice</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Sets the array slice values.
<a href="#"><u>SetValue</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Sets the array item value.
<a href="#"><u>WriteArray</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Writes all cached array values to the database.
<a href="#"><u>WriteArraySlice</u></a> (inherited from <a href="#"><u>TCustomIBCArra</u></a> )	Writes cached array slice.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.4.2 Properties

Properties of the **TIBCArray** class.

For a complete list of the **TIBCArray** class members, see the [TIBCArray Members](#) topic.

### Public

Name	Description
<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">ArrayDimensions</a> (inherited from <a href="#">TCustomIBCArray</a> )	Contains the array dimensions count.
<a href="#">ArrayHighBound</a> (inherited from <a href="#">TCustomIBCArray</a> )	Used to get or set the upper boundary of the defined dimension subscript.
<a href="#">ArrayID</a> (inherited from <a href="#">TCustomIBCArray</a> )	Contains the array ID.
<a href="#">ArrayLowBound</a> (inherited from <a href="#">TCustomIBCArray</a> )	Used to get or set the lower boundary of the defined dimension subscript.
<a href="#">ArraySize</a> (inherited from <a href="#">TCustomIBCArray</a> )	Used to determine the size of the whole array data in bytes.
<a href="#">Assign</a> (inherited from <a href="#">TCustomIBCArray</a> )	Copies the Source object content to the current one.
<a href="#">AsString</a> (inherited from <a href="#">TCustomIBCArray</a> )	Used to return array as string.
<a href="#">Cached</a> (inherited from <a href="#">TCustomIBCArray</a> )	Indicates whether to cache array data on the client side.
<a href="#">CachedDimensions</a> (inherited from <a href="#">TCustomIBCArray</a> )	Contains cached array dimensions count.
<a href="#">CachedHighBound</a> (inherited from <a href="#">TCustomIBCArray</a> )	Used to get the upper boundary of defined dimension subscript of cached array elements.
<a href="#">CachedLowBound</a> (inherited from <a href="#">TCustomIBCArray</a> )	Used to get the lower boundary of the defined dimension subscript of cached array elements.
<a href="#">CachedSize</a> (inherited from <a href="#">TCustomIBCArray</a> )	Used to get the cached array data size in bytes.
<a href="#">ClearArray</a> (inherited from <a href="#">TCustomIBCArray</a> )	Clears all array values on the server if Cached is set to False.
<a href="#">ColumnName</a> (inherited from <a href="#">TCustomIBCArray</a> )	Used to get or set the name of the table column that has array type.

<a href="#"><u>CreateTemporaryArray</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Creates a temporary array on the InterBase server.
<a href="#"><u>DbHandle</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Contains the handle of a database where the array is stored.
<a href="#"><u>GetArrayInfo</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Used to get array descriptor.
<a href="#"><u>GetItemAsDateTime</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Reads the value of an array item into an object or variable of the TDateTime type.
<a href="#"><u>GetItemAsFloat</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Reads the value of an array item into a floating-point number.
<a href="#"><u>GetItemAsInteger</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Reads the value of an array item into a integer.
<a href="#"><u>GetItemAsSmallInt</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Reads the value of an array item into a short integer.
<a href="#"><u>GetItemAsString</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Reads the value of an array item into a string.
<a href="#"><u>GetItemAsWideString</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Reads the value of an array item into a WideString.
<a href="#"><u>GetItemsSlice</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Returns the array slice items' values.
<a href="#"><u>GetValue</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Returns the array item value.
<a href="#"><u>IsNull</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Used to define whether the array field in the database is null.
<a href="#"><u>Items</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Used to get or set array items.
<a href="#"><u>ItemScale</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Used to get or set the scale for array items for the NUMERIC and DECIMAL datatypes.
<a href="#"><u>ItemSize</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Contains the size of an array item.
<a href="#"><u>ItemType</u></a>	Used to return the type of an array element.
<a href="#"><u>Modified</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Used to indicate if the modifications done in cache were saved to the database.
<a href="#"><u>ReadArray</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Reads an array from the database to memory.
<a href="#"><u>ReadArrayItem</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Reads the array item specified by indices from the database to memory.
<a href="#"><u>ReadArraySlice</u></a> (inherited from <a href="#"><u>TCustomIBCArray</u></a> )	Reads array slice from the database to memory.

<a href="#">RefCount</a> (inherited from <a href="#">TSharedObject</a> )	Used to return the count of reference to a TSharedObject object.
<a href="#">Release</a> (inherited from <a href="#">TSharedObject</a> )	Decrements the reference count.
<a href="#">SetItemAsDateTime</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Assigns the TDateTime value to the contents of an array item.
<a href="#">SetItemAsFloat</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Assigns floating-point value to the contents of an array item.
<a href="#">SetItemAsInteger</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Assigns integer value to the contents of an array item.
<a href="#">SetItemAsSmallInt</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Assigns short integer value to the contents of an array item.
<a href="#">SetItemAsString</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Assigns string value to the contents of an array item.
<a href="#">SetItemAsWideString</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Assigns WideString value to the contents of an array item.
<a href="#">SetItemsSlice</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Sets the array slice values.
<a href="#">SetItemValue</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Sets the array item value.
<a href="#">TableName</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Used to set the name of the table containing an array field.
<a href="#">TrHandle</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Contains the handle of the transaction in which the array is read or written.
<a href="#">WriteArray</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Writes all cached array values to the database.
<a href="#">WriteArraySlice</a> (inherited from <a href="#">TCustomIBCArrary</a> )	Writes cached array slice.

**See Also**

- [TIBCArrary Class](#)
- [TIBCArrary Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

17.13.1.4.2.1 ItemType Property

Used to return the type of an array element.

**Class**

[TIBCArrary](#)

**Syntax**

```
property ItemType: TFieldType;
```

**Remarks**

Read the ItemType property to return the type of an array element.

---

© 1997-2013 Devart. All Rights Reserved.

**17.13.1.5 TIBArrayField Class**

A class encapsulating the fundamental behavior common to the InterBase array fields.

For a list of all members of this type, see [TIBArrayField](#) members.

**Unit**

[IBC](#)

**Syntax**

```
TIBArrayField = class(TField);
```

**Remarks**

TIBArrayField encapsulates the fundamental behavior common to the InterBase array fields. The InterBase array fields can contain multiple data items of the same datatypes. TIBArrayField introduces new AsArray property for accessing array data.

**See Also**

- [TIBArray](#)
- 

© 1997-2013 Devart. All Rights Reserved.

**17.13.1.5.1 Members**

[TIBArrayField](#) class overview.

**Properties**

Name	Description
<a href="#">AsArray</a>	Used to specify the value of the field when it represents the value of the InterBase array type.

---

© 1997-2013 Devart. All Rights Reserved.

**17.13.1.5.2 Properties**

Properties of the **TIBArrayField** class.

For a complete list of the **TIBArrayField** class members, see the [TIBArrayField Members](#) topic.

**Public**



Name	Description
<a href="#">AsArray</a>	Used to specify the value of the field when it represents the value of the InterBase array type.

**See Also**

- [TIBCArryField Class](#)
- [TIBCArryField Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.5.2.1 AsArray Property

Used to specify the value of the field when it represents the value of the InterBase array type.

**Class**

[TIBCArryField](#)

**Syntax**

```
property AsArray: TIBCArry;
```

**Remarks**

Use the AsArray property to specify the value of the field when it represents the value of the InterBase array type.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6 TIBCCConnection Class

A component for setting and controlling connections to an InterBase database. For a list of all members of this type, see [TIBCCConnection](#) members.

**Unit**

[IBC](#)

**Syntax**

```
TIBCCConnection = class(TCustomDAConnection);
```

**Remarks**

The TIBCCConnection component is used to maintain connection to the InterBase database. After setting the Username, Password, and Database properties, you can establish a connection to the database by calling the Open method or setting the Connected property to True.

The TIBCCConnection component contains internal transaction component that is accessible through the [TIBCCConnection.DefaultTransaction](#) property. It is possible to create applications without manual adding TIBCTransaction components. But you can use external transaction components instead of internal transaction. To do it create a TIBCTransaction component and assign the TIBCCConnection.DefaultTransaction property to this transaction. If you need to restore internal

transaction - just reset the `TIBConnection.DefaultTransaction` property to nil. All components which are dedicated to perform data access, such as `TIBCQuery`, `TIBCSQL`, `TIBScript`, must have their `Connection` property assigned with one of the `TIBConnection` instances.

## Example

An example of `TIBConnection` creating and connecting is given below.

```
var
    Connection1 : TIBConnection;
...
procedure TForm1.Button1Click(Sender: TObject);
begin
    Connection1 := TIBConnection.Create(Self);
    Connection1.Database := 'C:\IBDatabases\SampleDatabase.gdb';
    Connection1.Server:='Server';
    Connection1.Options.Protocol:=TCP;
    Connection1.Connect;
end;
```

## Inheritance Hierarchy

[TCustomDAConnection](#)  
**TIBConnection**

## See Also

- [TIBConnection.DefaultTransaction](#)
- [TCustomIBCDDataSet.Connection](#)
- [TIBCSQL.Connection](#)

© 1997-2013 Devart. All Rights Reserved.

17.13.1.6.1 Members

[TIBConnection](#) class overview.

## Properties

Name	Description
<a href="#">AutoCommit</a>	Used to permit or prevent permanent updates, insertions, and deletions of data associated with the current transaction against the database server.
<a href="#">ClientLibrary</a>	Used to set or get the client library location.
<a href="#">ConnectDialog</a> (inherited from <a href="#">TCustomDAConnection</a> )	Allows to link a <a href="#">TCustomConnectDialog</a> component.

<a href="#">Connected</a>	Used to establish a database connection.
<a href="#">ConnectPrompt</a>	Used to provide login support.
<a href="#">ConvertEOL</a> (inherited from <a href="#">TCustomDAConnection</a> )	Allows customizing line breaks in string fields and parameters.
<a href="#">Database</a>	Used to set the name of the database to associate with TIBCCConnection component.
<a href="#">DatabaseInfo</a>	Used to get information about the connected database.
<a href="#">DBSQLDialect</a>	Shows the database SQL dialect.
<a href="#">Debug</a>	Used to display SQL statements being executed with their parameter values and data types.
<a href="#">DefaultTransaction</a>	Used to access default database connection transaction.
<a href="#">Handle</a>	Used to make calls directly to the InterBase API.
<a href="#">InTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )	Indicates whether the transaction is active.
<a href="#">LastError</a>	Used to get the error code which resulted from the previous call to the InterBase server.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomDAConnection</a> )	Specifies whether a login dialog appears immediately before opening a new connection.
<a href="#">Options</a>	Used to specify the behaviour of the TIBCCConnection object.
<a href="#">Params</a>	Used to specify the connection parameters.
<a href="#">Password</a>	Used to specify the password for a connection.
<a href="#">Pooling</a> (inherited from <a href="#">TCustomDAConnection</a> )	Enables or disables using connection pool.
<a href="#">PoolingOptions</a> (inherited from <a href="#">TCustomDAConnection</a> )	Specifies the behaviour of connection pool.
<a href="#">Server</a>	Used to supply the server name to handle server's request for a login.

<a href="#">SQL</a>	Used to execute any SQL statement.
<a href="#">SQLDialect</a>	Used to set or return SQL Dialect used by InterBase client.
<a href="#">TransactionCount</a>	Used to return the number of transactions currently associated with this TIBCCConnection component.
<a href="#">Transactions</a>	Used to specify a transaction for the given index.
<a href="#">Username</a>	Used to provide a user name.

## Methods

Name	Description
<a href="#">AddTransaction</a>	Associates a TIBCTransaction component with the database component.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TCustomDAConnection</a> )	Overloaded. Applies changes in datasets.
<a href="#">AssignConnect</a>	Shares database connection between the TIBCCConnection components.
<a href="#">Commit</a> (inherited from <a href="#">TCustomDAConnection</a> )	Commits current transaction.
<a href="#">CommitRetaining</a>	Stores to the database server all changes of data associated with the default database transaction permanently and then retains the transaction context.
<a href="#">Connect</a> (inherited from <a href="#">TCustomDAConnection</a> )	Establishes a connection to the server.
<a href="#">CreateDatabase</a>	Creates an InterBase database.
<a href="#">CreateDataSet</a> (inherited from <a href="#">TCustomDAConnection</a> )	Creates a dataset component.
<a href="#">CreateSQL</a> (inherited from <a href="#">TCustomDAConnection</a> )	Creates a component for queries execution.
<a href="#">Disconnect</a> (inherited from <a href="#">TCustomDAConnection</a> )	Performs disconnect.
<a href="#">DropDatabase</a>	Used to drop database and remove database file from server.

<a href="#">ExecProc</a> (inherited from <a href="#">TCustomDAConnection</a> )	Allows to execute stored procedure or function providing its name and parameters.
<a href="#">ExecProcEx</a> (inherited from <a href="#">TCustomDAConnection</a> )	Allows to execute a stored procedure or function.
<a href="#">ExecSQL</a> (inherited from <a href="#">TCustomDAConnection</a> )	Executes a SQL statement with parameters.
<a href="#">ExecSQLEx</a> (inherited from <a href="#">TCustomDAConnection</a> )	Executes any SQL statement outside the TQuery or TSQL components.
<a href="#">FindDefaultTransaction</a>	Returns the default transaction for the database.
<a href="#">GetDatabaseNames</a> (inherited from <a href="#">TCustomDAConnection</a> )	Returns a database list from the server.
<a href="#">GetStoredProcNames</a> (inherited from <a href="#">TCustomDAConnection</a> )	Returns a list of stored procedures from the server.
<a href="#">GetTableNames</a> (inherited from <a href="#">TCustomDAConnection</a> )	Provides a list of available tables names.
<a href="#">MonitorMessage</a> (inherited from <a href="#">TCustomDAConnection</a> )	Sends a specified message through the <a href="#">TCustomDASQLMonitor</a> component.
<a href="#">ParamByName</a>	POrovides access to the OUT parameters and their values after processing SQL statement.
<a href="#">RemoveFromPool</a> (inherited from <a href="#">TCustomDAConnection</a> )	Marks the connection that should not be returned to the pool after disconnect.
<a href="#">RemoveTransaction</a>	Disassociates a specified transaction from the database.
<a href="#">Rollback</a> (inherited from <a href="#">TCustomDAConnection</a> )	Discards all current data changes and ends transaction.
<a href="#">RollbackRetaining</a>	Rolls back all changes of data associated with the default database transaction to the database server and then retains the transaction context.
<a href="#">StartTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )	Begins a new user transaction.

## Events

Name	Description
<a href="#">OnConnectionLost</a> (inherited from <a href="#">TCustomDAConnection</a> )	This event occurs when connection was lost.

[OnError](#) (inherited from [TCustomDAConnection](#)) This event occurs when an error has arisen in the connection.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6.2 Properties

Properties of the **TIBConnection** class.

For a complete list of the **TIBConnection** class members, see the [TIBConnection Members](#) topic.

### Public

Name	Description
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TCustomDAConnection</a> )	Overloaded. Applies changes in datasets.
<a href="#">Commit</a> (inherited from <a href="#">TCustomDAConnection</a> )	Commits current transaction.
<a href="#">Connect</a> (inherited from <a href="#">TCustomDAConnection</a> )	Establishes a connection to the server.
<a href="#">ConnectDialog</a> (inherited from <a href="#">TCustomDAConnection</a> )	Allows to link a <a href="#">TCustomConnectDialog</a> component.
<a href="#">ConvertEOL</a> (inherited from <a href="#">TCustomDAConnection</a> )	Allows customizing line breaks in string fields and parameters.
<a href="#">CreateDataSet</a> (inherited from <a href="#">TCustomDAConnection</a> )	Creates a dataset component.
<a href="#">CreateSQL</a> (inherited from <a href="#">TCustomDAConnection</a> )	Creates a component for queries execution.
<a href="#">DatabaseInfo</a>	Used to get information about the connected database.
<a href="#">DBSQLDialect</a>	Shows the database SQL dialect.
<a href="#">Disconnect</a> (inherited from <a href="#">TCustomDAConnection</a> )	Performs disconnect.
<a href="#">ExecProc</a> (inherited from <a href="#">TCustomDAConnection</a> )	Allows to execute stored procedure or function providing its name and parameters.
<a href="#">ExecProcEx</a> (inherited from <a href="#">TCustomDAConnection</a> )	Allows to execute a stored procedure or function.
<a href="#">ExecSQL</a> (inherited from <a href="#">TCustomDAConnection</a> )	Executes a SQL statement with parameters.
<a href="#">ExecSQLEx</a> (inherited from <a href="#">TCustomDAConnection</a> )	Executes any SQL statement outside the TQuery or TSQL components.

<a href="#">GetDatabaseNames</a> (inherited from <a href="#">TCustomDAConnection</a> )	Returns a database list from the server.
<a href="#">GetStoredProcNames</a> (inherited from <a href="#">TCustomDAConnection</a> )	Returns a list of stored procedures from the server.
<a href="#">GetTableNames</a> (inherited from <a href="#">TCustomDAConnection</a> )	Provides a list of available tables names.
<a href="#">Handle</a>	Used to make calls directly to the InterBase API.
<a href="#">InTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )	Indicates whether the transaction is active.
<a href="#">LastError</a>	Used to get the error code which resulted from the previous call to the InterBase server.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomDAConnection</a> )	Specifies whether a login dialog appears immediately before opening a new connection.
<a href="#">MonitorMessage</a> (inherited from <a href="#">TCustomDAConnection</a> )	Sends a specified message through the <a href="#">TCustomDASQLMonitor</a> component.
<a href="#">OnConnectionLost</a> (inherited from <a href="#">TCustomDAConnection</a> )	This event occurs when connection was lost.
<a href="#">OnError</a> (inherited from <a href="#">TCustomDAConnection</a> )	This event occurs when an error has arisen in the connection.
<a href="#">Pooling</a> (inherited from <a href="#">TCustomDAConnection</a> )	Enables or disables using connection pool.
<a href="#">PoolingOptions</a> (inherited from <a href="#">TCustomDAConnection</a> )	Specifies the behaviour of connection pool.
<a href="#">RemoveFromPool</a> (inherited from <a href="#">TCustomDAConnection</a> )	Marks the connection that should not be returned to the pool after disconnect.
<a href="#">Rollback</a> (inherited from <a href="#">TCustomDAConnection</a> )	Discards all current data changes and ends transaction.
<a href="#">SQL</a>	Used to execute any SQL statement.
<a href="#">StartTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )	Begins a new user transaction.
<a href="#">TransactionCount</a>	Used to return the number of transactions currently associated with this TIBCCConnection component.
<a href="#">Transactions</a>	Used to specify a transaction for the given index.

## Published

Name	Description
<a href="#">AutoCommit</a>	Used to permit or prevent permanent updates, insertions, and deletions of data associated with the current transaction against the database server.
<a href="#">ClientLibrary</a>	Used to set or get the client library location.
<a href="#">Connected</a>	Used to establish a database connection.
<a href="#">ConnectPrompt</a>	Used to provide login support.
<a href="#">Database</a>	Used to set the name of the database to associate with TIBCCConnection component.
<a href="#">Debug</a>	Used to display SQL statements being executed with their parameter values and data types.
<a href="#">DefaultTransaction</a>	Used to access default database connection transaction.
<a href="#">Options</a>	Used to specify the behaviour of the TIBCCConnection object.
<a href="#">Params</a>	Used to specify the connection parameters.
<a href="#">Password</a>	Used to specify the password for a connection.
<a href="#">Server</a>	Used to supply the server name to handle server's request for a login.
<a href="#">SQLDialect</a>	Used to set or return SQL Dialect used by InterBase client.
<a href="#">Username</a>	Used to provide a user name.

**See Also**

- [TIBCCConnection Class](#)
- [TIBCCConnection Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

17.13.1.6.2.1 AutoCommit Property

Used to permit or prevent permanent updates, insertions, and deletions of data associated with the current transaction against the database server.



## Class

[TIBCCConnection](#)

## Syntax

```
property AutoCommit: boolean;
```

## Remarks

Use the AutoCommit property to permit or prevent permanent updates, insertions, and deletions of data associated with the current transaction against the database server without explicit calls to Commit or Rollback methods.

Set AutoCommit to True to permit implicit call to Commit method after every database access.

AutoCommit property in TIBCCConnection has higher precedence over the same properties in dataset components

The default value is True.

**Note:** The AutoCommit property in TIBCCConnection globally specifies whether all queries to modify database are implicitly committed or not. Components which descend from [TCustomDADataSet](#) and [TCustomDASQL](#) classes inherit their AutoCommit properties. This allows them to selectively specify their implicit transaction committing behavior after each data modifying access.

## Example

This procedure removes all records from Dept table and makes this change permanent.

```
procedure TForm1.DeleteClick(Sender: TObject);  
begin  
    IBCSQL.Connection := IBCCConnection;  
    IBCCConnection.AutoCommit := True;  
    IBCSQL.AutoCommit := False;  
    IBCSQL.SQL := 'DELETE FROM Dept';  
    IBCSQL.Execute;           // delete all records, commit is not performed  
    IBCCConnection.Rollback; // restore deleted records  
    IBCCConnection.AutoCommit := False;  
    IBCSQL.AutoCommit := True;  
    IBCSQL.SQL := 'DELETE FROM Dept';  
    IBCSQL.Execute;           // delete all records, commit is not performed  
    IBCCConnection.Rollback; // restore deleted records  
    IBCCConnection.AutoCommit := True;  
    IBCSQL.AutoCommit := True;  
    IBCSQL.SQL := 'DELETE FROM Dept';  
    IBCSQL.Execute;           // delete all records, commit is performed  
    IBCCConnection.Rollback; // couldn't restore deleted records  
end;
```

## See Also

- 

[TCustomDAConnection.Commit](#)

## 17.13.1.6.2.2 ClientLibrary Property

Used to set or get the client library location.

**Class**

[TIBCCConnection](#)

**Syntax**

```
property ClientLibrary: string;
```

**Remarks**

Use the ClientLibrary property to set or get the client library location.

**Note** : If you are using any .NET Delphi version (from Delphi 2005 to Delphi 2007), you can use only the gds32.dll library name. To work with the Firebird client library (fbclient.dll), you should rename it to gds32.dll.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.6.2.3 Connected Property

Used to establish a database connection.

**Class**

[TIBCCConnection](#)

**Syntax**

```
property Connected stored IsConnectedStored;
```

**Remarks**

Indicates whether the database connection is active. Set this property to True to establish a database connection. Setting this property to False will close a connection.

**See Also**

- [TCustomDAConnection.Connect](#)
  - [TCustomDAConnection.Disconnect](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.6.2.4 ConnectPrompt Property

Used to provide login support.

**Class**

[TIBCCConnection](#)

**Syntax**

```
property ConnectPrompt: boolean stored False default True;
```

**Remarks**

Set ConnectPrompt to True to provide login support when establishing a connection. When ConnectPrompt is True, a dialog appears to prompt a user for a name and a password.

When ConnectPrompt is False, an application must supply user name and password values programmatically.

**Warning:** Storing hard-coded user name and password entries as property values or in code for an OnLogin event handler can compromise the server security.

## See Also

- [Password](#)
- [Server](#)
- [Username](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.6.2.5 Database Property

Used to set the name of the database to associate with TIBCCConnection component.

## Class

[TIBCCConnection](#)

## Syntax

```
property Database: string;
```

## Remarks

Use the Database property to set the name of the database to associate with TIBCCConnection component. For local InterBase databases it is a filename. For remote databases use following syntax.

To connect to an InterBase database on a remote server using TCP/IP the syntax is <server name>:<filename>.

To connect to an InterBase database on a remote server using NetBEUI, the syntax is: \\<server name>\<filename>.

To connect to an InterBase database on a remote server using SPX, the syntax is: <server name>@<filename>.

**Note:** You can set connection parameters not only by Database property. You can use Server property to set server name and Options property to set connection protocol. If Database property contains local filename and server property is empty, it will be a local connection. If Server property is not empty, Server and Options properties will be used for the connection. If both Database and Server property contain server name and they are the same, server name and connection protocol will be taken from the Database property. If they are not the same, the server name and connection protocol will be taken from Server and Options properties and entire string of Database property will be processed as a database name without server name. In that case you may encounter errors.

## See Also

- [Options](#)
  - [Server](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6.2.6 DatabaseInfo Property

Used to get information about the connected database.

#### Class

[TIBConnection](#)

#### Syntax

```
property DatabaseInfo: TGDSDatabaseInfo;
```

#### Remarks

Use the DatabaseInfo property to get information about the connected database. You should explicitly add the [IBCClasses](#) unit to the 'uses' list to use this property.

#### See Also

- [TGDSDatabaseInfo](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6.2.7 DBSQLDialect Property

Shows the database SQL dialect.

#### Class

[TIBConnection](#)

#### Syntax

```
property DBSQLDialect: Integer;
```

#### Remarks

The DBSQLDialect property is used to show the database SQL dialect.

#### See Also

- [SQLDialect](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6.2.8 Debug Property

Used to display SQL statements being executed with their parameter values and data types.

#### Class

[TIBCCConnection](#)**Syntax**

```
property Debug: boolean;
```

**Remarks**

Set the Debug property to True to display SQL statements being executed with their parameter values and data types.

**Note:** To use this property you should explicitly include unit IBDACVcl (IBDACClx under Linux) to your project.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.6.2.9 DefaultTransaction Property

Used to access default database connection transaction.

**Class**[TIBCCConnection](#)**Syntax**

```
property DefaultTransaction: TIBCTransaction;
```

**Remarks**

Use the DefaultTransaction property to access default database connection transaction. By default this is internal connection transaction. You can set it to external transaction component. To restore internal transaction set this property to nil.

**See Also**

- [Transactions](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.6.2.10 Handle Property

Used to make calls directly to the InterBase API.

**Class**[TIBCCConnection](#)**Syntax**

```
property Handle: TISC_DB_HANDLE;
```

**Remarks**

Use the Handle property to make calls directly to the InterBase API. Many of the InterBase API functions require a database handle as an argument.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.6.2.11 LastError Property

Used to get the error code which resulted from the previous call to the InterBase server.

**Class**

[TIBConnection](#)

**Syntax**

```
property LastError: integer;
```

**Remarks**

Use the LastError property to get the error code which resulted from the previous call to the InterBase server.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.6.2.12 Options Property

Used to specify the behaviour of the TIBConnection object.

**Class**

[TIBConnection](#)

**Syntax**

```
property Options: TIBConnectionOptions;
```

**Remarks**

Set properties of Options to specify the behaviour of a TIBConnection object. Descriptions of all options are in the table below.

Option Name	Description
<a href="#">CharLength</a>	Used to specify the size in bytes of a single character.
<a href="#">Charset</a>	Used to set character set that IBDAC uses to read and write character data.
<a href="#">EnableBCD</a>	Used to enable currency type.
<a href="#">EnableFMTBCD</a>	Used to enable using FMTBCD instead of float for large integer numbers to keep precision.
<a href="#">EnableMemos</a>	Used to enable creating TMemoField and TWideMemoField for BLOB fields with subtype 1.
<a href="#">Protocol</a>	Used to specify the Network protocol of connection with InterBase server.
<a href="#">Role</a>	Used to specify the InterBase connection role.
<a href="#">UseUnicode</a>	Used to enable or disable Unicode support.

## See Also

- [TCustomDAConnection.Options](#)
- [Unicode Character Data](#)

©

1997-2013 Devart. All Rights Reserved.

### 17.13.1.6.2.13 Params Property

Used to specify the connection parameters.

## Class

[TIBCCConnection](#)

## Syntax

```
property Params: TStrings;
```

## Remarks

Use the Params property to specify the connection parameters, such as user name, password, role etc. For example: user name=sysdba password=masterkey sql role name=role1 lc ctype=WIN1252

However, note that TIBCCConnection has separate Username and Password properties. Therefore, to specify a user name and a password for connection, you should use either Username and Password properties or Params, but not both.

If you are using the Params property for creating a new database with the CreateDatabase function, you should set params like it is shown below: USER "SYSDBA" PASSWORD "masterkey" PAGE SIZE 4096

## See Also

- [CreateDatabase](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.6.2.14 Password Property

Used to specify the password for a connection.

## Class

[TIBCCConnection](#)

## Syntax

```
property Password: string;
```

**Remarks**

Use the Password property to specify a password for connection.  
When property is being changed TIBCCConnection calls the Disconnect method.

**See Also**

- [Username](#)
  - [Server](#)
  - [TCustomDAConnection.Connect](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.6.2.15 Server Property

Used to supply the server name to handle server's request for a login.

**Class**

[TIBCCConnection](#)

**Syntax**

```
property Server: string;
```

**Remarks**

Use the Server property to supply the server name to handle server's request for a login.  
When property is being changed TIBCCConnection calls the Disconnect method.

**See Also**

- [Username](#)
  - [Password](#)
  - [TCustomDAConnection.Connect](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.6.2.16 SQL Property

Used to execute any SQL statement.

**Class**

[TIBCCConnection](#)

**Syntax**

```
property SQL: TIBCSQL;
```

**Remarks**

You can use the embedded TIBCSQL object to execute any SQL statement.



## See Also

- [TIBCSQL](#)
- [TCustomDAConnection.ExecSQL](#)
- [TCustomDAConnection.ExecSQLEx](#)
- [ParamByName](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.6.2.17 SQLDialect Property

Used to set or return SQL Dialect used by InterBase client.

## Class

[TIBConnection](#)

## Syntax

```
property SQLDialect: Integer default 3;
```

## Remarks

Use the SQLDialect property to set or return SQL Dialect used by InterBase client. The SQLDialect property cannot be set to a value greater than the database SQL dialect when the connection is active. If the connection is inactive, the SQLDialect property will be downgraded to match the database SQL dialect.

## See Also

- [DBSQLDialect](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.6.2.18 TransactionCount Property

Used to return the number of transactions currently associated with this TIBConnection component.

## Class

[TIBConnection](#)

## Syntax

```
property TransactionCount: integer;
```

## Remarks

Use the TransactionCount property to return the number of transactions currently associated with this TIBConnection component.

## See Also

- [Transactions](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6.2.19 Transactions Property(Indexer)

Used to specify a transaction for the given index.

### Class

[TIBConnection](#)

### Syntax

```
property Transactions[Index: Integer]: TIBTransaction;
```

### Parameters

*Index*

Holds the specified index.

### Remarks

Use the Transactions property to specify a transaction for the given index.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6.2.20 Username Property

Used to provide a user name.

### Class

[TIBConnection](#)

### Syntax

```
property Username: string;
```

### Remarks

Use the Username property to supply the user name to handle server's request for a login.

When property is being changed TIBConnection calls Disconnect method.

### See Also

- [Password](#)
  - [Server](#)
  - [TCustomDAConnection.Connect](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6.3 Methods

Methods of the **TIBConnection** class.

For a complete list of the **TIBConnection** class members, see the [TIBConnection Members](#) topic.

**Public**

Name	Description
<a href="#">AddTransaction</a>	Associates a
	TIBCTransaction component
	with the database
	component.
<a href="#">ApplyUpdates</a> (inherited from	Overloaded. Applies changes
<a href="#">TCustomDAConnection</a> )	in datasets.
<a href="#">AssignConnect</a>	Shares database connection
	between the TIBConnection
	components.
<a href="#">Commit</a> (inherited from <a href="#">TCustomDAConnection</a> )	Commits current
	transaction.
<a href="#">CommitRetaining</a>	Stores to the database
	server all changes of data
	associated with the default
	database transaction
	permanently and then
	retains the transaction
	context.
<a href="#">Connect</a> (inherited from <a href="#">TCustomDAConnection</a>	Establishes a connection to
)	the server.
<a href="#">ConnectDialog</a> (inherited from	Allows to link a
<a href="#">TCustomDAConnection</a> )	<a href="#">TCustomConnectDialog</a>
	component.
<a href="#">ConvertEOL</a> (inherited from	Allows customi ing line
<a href="#">TCustomDAConnection</a> )	breaks in string fields and
	parameters.
<a href="#">CreateDatabase</a>	Creates an InterBase
	database.
<a href="#">CreateDataSet</a> (inherited from	Creates a dataset
<a href="#">TCustomDAConnection</a> )	component.
<a href="#">CreateSQL</a> (inherited from	Creates a component for
<a href="#">TCustomDAConnection</a> )	queries execution.
<a href="#">Disconnect</a> (inherited from	Performs disconnect.
<a href="#">TCustomDAConnection</a> )	
<a href="#">DropDatabase</a>	Used to drop database and
	remove database file from
	server.
<a href="#">ExecProc</a> (inherited from <a href="#">TCustomDAConnection</a>	Allows to execute stored
)	procedure or function
	providing its name and
	parameters.
<a href="#">ExecProcEx</a> (inherited from	Allows to execute a stored
<a href="#">TCustomDAConnection</a> )	procedure or function.
<a href="#">ExecSQL</a> (inherited from <a href="#">TCustomDAConnection</a>	Executes a SQL statement
)	with parameters.

<a href="#"><u>ExecSQLEx</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Executes any SQL statement outside the TQuery or TSQL components.
<a href="#"><u>FindDefaultTransaction</u></a>	Returns the default transaction for the database.
<a href="#"><u>GetDatabaseNames</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Returns a database list from the server.
<a href="#"><u>GetStoredProcNames</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Returns a list of stored procedures from the server.
<a href="#"><u>GetTableNames</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Provides a list of available tables names.
<a href="#"><u>InTransaction</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Indicates whether the transaction is active.
<a href="#"><u>LoginPrompt</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Specifies whether a login dialog appears immediately before opening a new connection.
<a href="#"><u>MonitorMessage</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Sends a specified message through the <a href="#"><u>TCustomDASQLMonitor</u></a> component.
<a href="#"><u>OnConnectionLost</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	This event occurs when connection was lost.
<a href="#"><u>OnError</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	This event occurs when an error has arisen in the connection.
<a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Specifies the connection behavior.
<a href="#"><u>ParamByName</u></a>	POrovides access to the OUT parameters and their values after processing SQL statement.
<a href="#"><u>Password</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Serves to supply a password for login.
<a href="#"><u>Pooling</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Enables or disables using connection pool.
<a href="#"><u>PoolingOptions</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Specifies the behaviour of connection pool.
<a href="#"><u>RemoveFromPool</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Marks the connection that should not be returned to the pool after disconnect.
<a href="#"><u>RemoveTransaction</u></a>	Disassociates a specified transaction from the database.
<a href="#"><u>Rollback</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )	Discards all current data changes and ends transaction.

[RollbackRetaining](#)

Rolls back all changes of data associated with the default database transaction to the database server and then retains the transaction context.

[Server](#) (inherited from [TCustomDAConnection](#))

Serves to supply the server name for login.

[StartTransaction](#) (inherited from [TCustomDAConnection](#))

Begins a new user transaction.

[Username](#) (inherited from [TCustomDAConnection](#))

Used to supply a user name for login.

**See Also**

- [TIBConnection Class](#)
- [TIBConnection Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6.3.1 AddTransaction Method

Associates a TIBCTransaction component with the database component.

**Class**

[TIBConnection](#)

**Syntax**

```
function AddTransaction(TR: TIBCTransaction): Integer;
```

**Parameters**

*TR*

Holds the transaction that is being added.

**Return Value**

the index of the associated transaction in the transaction list.

**Remarks**

Use the AddTransaction method to associate a TIBCTransaction component with the database component. Returns the index of the associated transaction in the transaction list.

**See Also**

- [Transactions](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.6.3.2 AssignConnect Method

Shares database connection between the TIBCCConnection components.

**Class**

[TIBCCConnection](#)

**Syntax**

```
procedure AssignConnect (Source: TIBCCConnection) ;
```

**Parameters**

*Source*

Holds the source connection.

**Remarks**

Call the AssignConnect method to share database connection between the TIBCCConnection components.

AssignConnect assumes that Source parameter points to a preconnected database component and sets for this instance of TIBCCConnection Connected property to True. Note that AssignConnect doesn't make any references to the Source database. So before disconnecting parent TIBCCConnection call AssignConnect(nil) or Disconnect method for all assigned databases.

**Example**

```
IBCCConnection1.Connect;
IBCCConnection2.AssignConnect (IBCCConnection1);
// IBCCConnection2.Connected is True
IBCSQL.Connection := IBCCConnection2;
IBCSQL.Execute;
IBCCConnection2.AssignConnect (nil);
// IBCCConnection2.Connected is False
IBCCConnection1.Disconnect;
```

**See Also**

- 

[TCustomDAConnection.Connect](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.6.3.3 CommitRetaining Method

Stores to the database server all changes of data associated with the default database transaction permanently and then retains the transaction context.

**Class**

[TIBCCConnection](#)

**Syntax**

```
procedure CommitRetaining;
```

**Remarks**

Call the CommitRetaining method to store to the database server all changes of data associated with the default database transaction permanently and then retain the transaction context.

**See Also**

- [DefaultTransaction](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6.3.4 CreateDatabase Method

Creates an InterBase database.

**Class**

[TIBConnection](#)

**Syntax**

```
procedure CreateDatabase;
```

**Remarks**

Call the CreateDatabase method to create an InterBase database using [Params](#) as the rest of the CREATE DATABASE statement.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6.3.5 DropDatabase Method

Used to drop database and remove database file from server.

**Class**

[TIBConnection](#)

**Syntax**

```
procedure DropDatabase;
```

**Remarks**

Call the DropDatabase method to drop database and remove database file from server.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6.3.6 FindDefaultTransaction Method

Returns the default transaction for the database.

**Class**

[TIBConnection](#)

**Syntax**

```
function FindDefaultTransaction: TIBCTransaction;
```

**Return Value**

the default transaction for the database.

**Remarks**

Call the FindDefaultTransaction method to return the default transaction for the database.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.6.3.7 ParamByName Method

POrovides access to the OUT parameters and their values after processing SQL statement.

**Class**

[TIBCCConnection](#)

**Syntax**

```
function ParamByName(const Name: string): TIBCParam;
```

**Parameters**

*Name*

Holds the parameter name.

**Return Value**

the corresponding parameter.

**Remarks**

Call the ParamByName method to get access to the OUT parameters and their values after processing SQL statement with ExecSQL or stored procedure with ExecProc. Name should be equal to the parameter name as it occurred in SQL statement.

Implicitly calls the [TIBCSQL.ParamByName](#) function of [TIBCSQL](#).

**See Also**

- [SQL](#)
  - [TCustomDAConnection.ExecSQL](#)
  - [TCustomDAConnection.ExecSQLEx](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.6.3.8 RemoveTransaction Method

Disassociates a specified transaction from the database.

**Class**



[TIBCCConnection](#)**Syntax**

```
procedure RemoveTransaction (TR: TIBCTransaction) ;
```

**Parameters**

TR

**Remarks**

Call the RemoveTransaction method to disassociate a specified transaction from the database.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.6.3.9 RollbackRetaining Method

Rolls back all changes of data associated with the default database transaction to the database server and then retains the transaction context.

**Class**[TIBCCConnection](#)**Syntax**

```
procedure RollbackRetaining;
```

**Remarks**

Call the RollbackRetaining method to roll back all changes of data associated with the default database transaction to the database server and then retain the transaction context.

**See Also**

- [DefaultTransaction](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.7 TIBCCConnectionOptions Class

This class allows setting up the behaviour of the TIBCCConnection class.  
For a list of all members of this type, see [TIBCCConnectionOptions](#) members.

**Unit**[IBC](#)**Syntax**

```
TIBCCConnectionOptions = class (TDACConnectionOptions) ;
```

**Inheritance Hierarchy**[TDACConnectionOptions](#)**TIBCCConnectionOptions**

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.7.1 Members

[TIBConnectionOptions](#) class overview.

### Properties

Name	Description
<a href="#">CharLength</a>	Used to specify the size in bytes of a single character.
<a href="#">Charset</a>	Used to set character set that IBDAC uses to read and write character data.
<a href="#">DefaultSortType</a> (inherited from <a href="#">TDACConnectionOptions</a> )	Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the <a href="#">TMemDataSet.IndexFieldNames</a> property of a dataset.
<a href="#">DisconnectedMode</a> (inherited from <a href="#">TDACConnectionOptions</a> )	Used to open a connection only when needed for performing a server call and closes after performing the operation.
<a href="#">EnableBCD</a>	Used to enable currency type.
<a href="#">EnableFMTBCD</a>	Used to enable using FMTBCD instead of float for large integer numbers to keep precision.
<a href="#">EnableMemos</a>	Used to enable creating TMemoField and TWideMemoField for BLOB fields with subtype 1.
<a href="#">KeepDesignConnected</a> (inherited from <a href="#">TDACConnectionOptions</a> )	Used to prevent an application from establishing a connection at the time of startup.
<a href="#">LocalFailover</a> (inherited from <a href="#">TDACConnectionOptions</a> )	If True, the <a href="#">TCustomDACConnection.OnConnectionLost</a> event occurs and a failover operation can be performed after connection breaks.
<a href="#">Protocol</a>	Used to specify the Network protocol of connection with InterBase server.

[Role](#)

Used to specify the InterBase connection role.

[UseUnicode](#)

Used to enable or disable Unicode support.

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.7.2 Properties

Properties of the **TIBConnectionOptions** class.For a complete list of the **TIBConnectionOptions** class members, see the [TIBConnectionOptions Members](#) topic.**Public****Name****Description**[DefaultSortType](#) (inherited from [TDACConnectionOptions](#))Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the [TMemDataSet.IndexFieldNames](#) property of a dataset.[DisconnectedMode](#) (inherited from [TDACConnectionOptions](#))

Used to open a connection only when needed for performing a server call and closes after performing the operation.

[KeepDesignConnected](#) (inherited from [TDACConnectionOptions](#))

Used to prevent an application from establishing a connection at the time of startup.

[LocalFailover](#) (inherited from [TDACConnectionOptions](#))If True, the [TCustomDACConnection.OnConnectionLost](#) event occurs and a failover operation can be performed after connection breaks.**Published****Name****Description**[CharLength](#)

Used to specify the size in bytes of a single character.

[Charset](#)

Used to set character set that IBDAC uses to read and write character data.

[EnableBCD](#)

Used to enable currency type.

[EnableFMTBCD](#)

Used to enable using FMTBCD instead of float for large integer numbers to keep precision.

[EnableMemos](#)

Used to enable creating TMemoField and TWideMemoField for BLOB fields with subtype 1.

[Protocol](#)

Used to specify the Network protocol of connection with InterBase server.

[Role](#)

Used to specify the InterBase connection role.

[UseUnicode](#)

Used to enable or disable Unicode support.

**See Also**

- [TIBCCConnectionOptions Class](#)
  - [TIBCCConnectionOptions Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.7.2.1 CharLength Property

Used to specify the size in bytes of a single character.

**Class**[TIBCCConnectionOptions](#)**Syntax**

```
property CharLength: TIBCCCharLength default 0;
```

**Remarks**

Use the CharLength property to specify the size in bytes of a single character. Set this option with the number in range [0..6] to reflect InterBase support for the national languages. Setting CharLength to zero will instruct TIBCCConnection to interrogate InterBase server for the actual character length. The default value is 1.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.7.2.2 Charset Property

Used to set character set that IBDAC uses to read and write character data.

**Class**[TIBCCConnectionOptions](#)**Syntax**

```
property Charset: string;
```

**Remarks**

Use the Charset property to set character set that IBDAC uses to read and write character data.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.7.2.3 EnableBCD Property

Used to enable currency type.

**Class**

[TIBCCConnectionOptions](#)

**Syntax**

```
property EnableBCD: boolean;
```

**Remarks**

Use the EnableBCD property to enable currency type.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.7.2.4 EnableFMTBCD Property

Used to enable using FMTBCD instead of float for large integer numbers to keep precision.

**Class**

[TIBCCConnectionOptions](#)

**Syntax**

```
property EnableFMTBCD: boolean;
```

**Remarks**

Use the EnableFMTBCD property to enable using FMTBCD instead of float for large integer numbers to keep precision.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.7.2.5 EnableMemos Property

Used to enable creating TMemoField and TWideMemoField for BLOB fields with subtype 1.

**Class**

[TIBCCConnectionOptions](#)

**Syntax**

```
property EnableMemos: boolean default False;
```

**Remarks**

Use the EnableMemos property to enable creating TMemoField and TWideMemoField for BLOB fields with subtype 1.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.7.2.6 Protocol Property

Used to specify the Network protocol of connection with InterBase server.

##### **Class**

[TIBCCConnectionOptions](#)

##### **Syntax**

```
property Protocol: TIBCPProtocol default TCP;
```

##### **Remarks**

Use the Protocol property to specify the network protocol of connection with InterBase server. The default value is TCP.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.7.2.7 Role Property

Used to specify the InterBase connection role.

##### **Class**

[TIBCCConnectionOptions](#)

##### **Syntax**

```
property Role: string;
```

##### **Remarks**

Use the Role property to specify the InterBase connection role.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.7.2.8 UseUnicode Property

Used to enable or disable Unicode support.

##### **Class**

[TIBCCConnectionOptions](#)

##### **Syntax**

```
property UseUnicode: boolean default False;
```

##### **Remarks**

Use the UseUnicode property to enable or disable Unicode support. Affects on character data fetched from the server. When set to True all character data is stored as WideStrings, and TStringField is replaced with TWideStringField.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.8 TIBDataSetOptions Class

This class allows setting up the behaviour of the TIBDataSet class.  
For a list of all members of this type, see [TIBDataSetOptions](#) members.

#### Unit

[IBC](#)

#### Syntax

```
TIBDataSetOptions = class(TDADatasetOptions);
```

#### Inheritance Hierarchy

[TDADatasetOptions](#)

**TIBDataSetOptions**

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.8.1 Members

[TIBDataSetOptions](#) class overview.

#### Properties

Name	Description
<a href="#">AutoClose</a>	Used to for CustomIBDataSet to close cursor after fetching all rows.
<a href="#">AutoPrepare</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to execute automatic <a href="#">TCustomDADataset.Prepare</a> on the query execution.
<a href="#">BooleanDomainFields</a>	Used to create TBooleanField for fields that have domain of the integer data type, and the domain name contains 'BOOLEAN'.
<a href="#">CacheArrays</a>	Used to allocate local memory buffer to hold a copy of the array content.
<a href="#">CacheBlobs</a>	Used to allocate local memory buffer to hold a copy of the BLOB content.
<a href="#">CacheCalcFields</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to enable caching of the TField.Calculated and TField.Lookup fields.
<a href="#">ComplexArrayFields</a>	Used to store array fields as TIBCArraryField objects.

<a href="#">CompressBlobMode</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to store values of the BLOB fields in compressed form.
<a href="#">DefaultValues</a>	Used for TCustomIBCDataset to fill the DefaultExpression property of TField objects by the appropriate value.
<a href="#">DeferredArrayRead</a>	Used for fetching all InterBase array values when they are explicitly requested.
<a href="#">DeferredBlobRead</a>	Used for fetching all InterBase BLOB values when they are explicitly requested.
<a href="#">DescribeParams</a>	Used to specify whether to query <a href="#">TIBCPParam</a> properties from the server when executing the <a href="#">TCustomDADataset.Prepare</a> method.
<a href="#">DetailDelay</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.
<a href="#">FieldsAsString</a>	Used to treat all non-BLOB fields as being of string datatype.
<a href="#">FieldsOrigin</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.
<a href="#">FlatBuffers</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to control how a dataset treats data of the ftString and ftVarBytes fields.
<a href="#">FullRefresh</a>	Used to refresh fields from all tables of the query.
<a href="#">LocalMasterDetail</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.



<a href="#">LongStrings</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to represent string fields with the length that is greater than 255 as TStringField.
<a href="#">NumberRange</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.
<a href="#">QueryRecCount</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for TCustomDADataset to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.
<a href="#">QuoteNames</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.
<a href="#">RemoveOnRefresh</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for a dataset to locally remove a record that can not be found on the server.
<a href="#">RequiredFields</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.
<a href="#">ReturnParams</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to return the new value of fields to dataset after insert or update.
<a href="#">SetFieldsReadOnly</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.
<a href="#">StreamedBlobs</a>	Used to save all edited BLOBs as streamed and to handle the streamed
<a href="#">StrictUpdate</a>	Used for TCustomIBCDataset to raise an exception when the number of the updated or deleted records is not equal 1.
<a href="#">TrimFixedChar</a> (inherited from <a href="#">TDADatasetOptions</a> )	Specifies whether to discard all trailing spaces in the string fields of a dataset.

[UpdateAllFields](#) (inherited from [TDADatasetOptions](#))

Used to include all dataset fields in the generated UPDATE and INSERT statements.

[UpdateBatchSize](#) (inherited from [TDADatasetOptions](#))

Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.8.2 Properties

Properties of the **TIBCDatasetOptions** class.

For a complete list of the **TIBCDatasetOptions** class members, see the [TIBCDatasetOptions Members](#) topic.

#### Public

Name	Description
<a href="#">AutoPrepare</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to execute automatic <a href="#">TCustomDADataset.Prepare</a> on the query execution.
<a href="#">CacheCalcFields</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to enable caching of the TField.Calculated and TField.Lookup fields.
<a href="#">CompressBlobMode</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to store values of the BLOB fields in compressed form.
<a href="#">DetailDelay</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.
<a href="#">FieldsOrigin</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.
<a href="#">FlatBuffers</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to control how a dataset treats data of the ftString and ftVarBytes fields.
<a href="#">LocalMasterDetail</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.

<a href="#">LongStrings</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to represent string fields with the length that is greater than 255 as TStringField.
<a href="#">NumberRange</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.
<a href="#">QueryRecCount</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for TCustomDADataset to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.
<a href="#">QuoteNames</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.
<a href="#">RemoveOnRefresh</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for a dataset to locally remove a record that can not be found on the server.
<a href="#">RequiredFields</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.
<a href="#">ReturnParams</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to return the new value of fields to dataset after insert or update.
<a href="#">SetFieldsReadOnly</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.
<a href="#">TrimFixedChar</a> (inherited from <a href="#">TDADatasetOptions</a> )	Specifies whether to discard all trailing spaces in the string fields of a dataset.
<a href="#">UpdateAllFields</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to include all dataset fields in the generated UPDATE and INSERT statements.
<a href="#">UpdateBatchSize</a> (inherited from <a href="#">TDADatasetOptions</a> )	Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

**Published**

Name	Description
<a href="#">AutoClose</a>	Used to for CustomIBCDataset to close cursor after fetching all rows.
<a href="#">BooleanDomainFields</a>	Used to create TBooleanField for fields that have domain of the integer data type, and the domain name contains 'BOOLEAN'.
<a href="#">CacheArrays</a>	Used to allocate local memory buffer to hold a copy of the array content.
<a href="#">CacheBlobs</a>	Used to allocate local memory buffer to hold a copy of the BLOB content.
<a href="#">ComplexArrayFields</a>	Used to store array fields as TIBCArraryField objects.
<a href="#">DefaultValues</a>	Used for TCustomIBCDataset to fill the DefaultExpression property of TField objects by the appropriate value.
<a href="#">DeferredArrayRead</a>	Used for fetching all InterBase array values when they are explicitly requested.
<a href="#">DeferredBlobRead</a>	Used for fetching all InterBase BLOB values when they are explicitly requested.
<a href="#">DescribeParams</a>	Used to specify whether to query <a href="#">TIBCPParam</a> properties from the server when executing the <a href="#">TCustomDADataset.Prepare</a> method.
<a href="#">FieldsAsString</a>	Used to treat all non-BLOB fields as being of string datatype.
<a href="#">FullRefresh</a>	Used to refresh fields from all tables of the query.
<a href="#">StreamedBlobs</a>	Used to save all edited BLOBs as streamed and to handle the streamed

[StrictUpdate](#)

Used for TCustomIBCDataset to raise an exception when the number of the updated or deleted records is not equal 1.

**See Also**

- [TIBCDatasetOptions Class](#)
- [TIBCDatasetOptions Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.8.2.1 AutoClose Property

Used to for CustomIBCDataset to close cursor after fetching all rows.

**Class**

[TIBCDatasetOptions](#)

**Syntax**

```
property AutoClose: boolean default False;
```

**Remarks**

Use the AutoClose property for CustomIBCDataset to close cursor after fetching all rows. Allows to reduce the number of opened cursors on the server.

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.8.2.2 BooleanDomainFields Property

Used to create TBooleanField for fields that have domain of the integer data type, and the domain name contains 'BOOLEAN'.

**Class**

[TIBCDatasetOptions](#)

**Syntax**

```
property BooleanDomainFields: boolean default False;
```

**Remarks**

If the BooleanDomainFields property is set to True, TBooleanField objects are created for fields that have domain of the integer data type, and the domain name contains 'BOOLEAN'.

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.8.2.3 CacheArrays Property

Used to allocate local memory buffer to hold a copy of the array content.

**Class**

[TIBCDatasetOptions](#)**Syntax**

```
property CacheArrays: boolean default True;
```

**Remarks**

If the CacheArray property is set to True (the default value), then local memory buffer is allocated to hold a copy of the array content.

**Note:** This option does not make sense if DeferredArrayRead is set to False because all BLOB values are fetched to the dataset in that case.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.8.2.4 CacheBlobs Property

Used to allocate local memory buffer to hold a copy of the BLOB content.

**Class**[TIBCDatasetOptions](#)**Syntax**

```
property CacheBlobs: boolean default True;
```

**Remarks**

If the CacheBlobs property is set to True (the default value), then local memory buffer is allocated to hold a copy of the BLOB content.

**Note:** CacheBlobs option controls the way streamed BLOB objects are handled. If False, application can access streamed BLOB values on server side without caching BLOBs on client side. Only requested portions of data are fetched. Setting CacheBlobs to False may bring up the following benefits for time-critical applications: reduced traffic over the network since only required data are fetched, less memory is needed on the client side because returned record sets do not hold contents of BLOB fields. This feature is available only for streamed BLOBs and only if StreamedBlobs option is set to True. This option doesn't make sense if DeferredBlobRead is set to False because all BLOB values are fetched to the dataset in that case.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.8.2.5 ComplexArrayFields Property

Used to store array fields as TIBCArraryField objects.

**Class**[TIBCDatasetOptions](#)**Syntax**

```
property ComplexArrayFields: boolean default True;
```

**Remarks**

If the ComplexArrayFields property is set to False, any array field is stored as one

TIBArrayField object. If true and ObjectView is true, array items are stored hierarchically. If true and ObjectView is false, all array items are stored as sibling fields.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.8.2.6 DefaultValues Property

Used for TCustomIBCDataset to fill the DefaultExpression property of TField objects by the appropriate value.

##### Class

[TIBCDatasetOptions](#)

##### Syntax

```
property DefaultValues: boolean;
```

##### Remarks

If the DefaultValues property is set to True, TCustomIBCDataset fills the DefaultExpression property of TField objects by the appropriate value. Note that computed BLR fields are not detected and set to read-only if DefaultValues set to false.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.8.2.7 DeferredArrayRead Property

Used for fetching all InterBase array values when they are explicitly requested.

##### Class

[TIBCDatasetOptions](#)

##### Syntax

```
property DeferredArrayRead: boolean default True;
```

##### Remarks

If the DeferredArrayRead property is set to True, all InterBase array values are only fetched when they are explicitly requested. Otherwise entire record set with any array values is returned when dataset is opened. Whether array values are cached locally to be reused later or not is controlled by the CacheBlobs option.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.8.2.8 DeferredBlobRead Property

Used for fetching all InterBase BLOB values when they are explicitly requested.

##### Class

[TIBCDatasetOptions](#)

##### Syntax

```
property DeferredBlobRead: boolean default False;
```

**Remarks**

If the DeferredBlobRead property is set to True, all InterBase BLOB values are only fetched when they are explicitly requested. Otherwise entire record set with any BLOB values is returned when dataset is opened. Whether BLOB values are cached locally to be reused later or not is controlled by CacheBlobs option.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.8.2.9 DescribeParams Property

Used to specify whether to query [TIBCPParam](#) properties from the server when executing the [TCustomDADataset.Prepare](#) method.

**Class**

[TIBCDatasetOptions](#)

**Syntax**

```
property DescribeParams: boolean default False;
```

**Remarks**

Specifies whether to query [TIBCPParam](#) properties (Name, ParamType, DataType, Size, TableName) from the server when executing the [TCustomDADataset.Prepare](#) method. The default value is False.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.8.2.10 FieldsAsString Property

Used to treat all non-BLOB fields as being of string datatype.

**Class**

[TIBCDatasetOptions](#)

**Syntax**

```
property FieldsAsString: boolean default False;
```

**Remarks**

If the FieldsAsString property is set to True, then all non-BLOB fields are treated as being of string datatype.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.8.2.11 FullRefresh Property

Used to refresh fields from all tables of the query.

**Class**

[TIBCDatasetOptions](#)

**Syntax**

```
property FullRefresh: boolean;
```



## Remarks

Use the FullRefresh property to refresh fields from all tables of the query.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.8.2.12 StreamedBlobs Property

Used to save all edited BLOBs as streamed and to handle the streamed

## Class

[TIBCDatasetOptions](#)

## Syntax

```
property StreamedBlobs: boolean default False;
```

## Remarks

If the StreamedBlobs property is set to True, then all edited BLOBs are saved as streamed BLOBs and all streamed BLOBs are handled as streamed. Otherwise streamed BLOBs are handled as usual segmented BLOBs and all edited BLOBs are saved as segmented BLOBs. For more information on BLOBs see [BLOB Data Types](#). Setting this option to True allows to use benefits of the CacheBlobs option.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.8.2.13 StrictUpdate Property

Used for TCustomIBCDataset to raise an exception when the number of the updated or deleted records is not equal 1.

## Class

[TIBCDatasetOptions](#)

## Syntax

```
property StrictUpdate: boolean;
```

## Remarks

TCustomIBCDataset raises exception when the number of the updated or deleted records is not equal 1. Setting this option also causes an exception if the RefreshRecord procedure returns more than one record. The exception does not occur when you use non-SQL block. The default value is True. If False, the AffectedRows property is not calculated and becomes equal zero. This can improve performance of query executing, so if you need to execute many data updating statements at once and you don't mind affected rows count, set this property to False.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.9 TIBCDatasource Class

TIBCDatasource provides an interface between an IBDAC dataset components and data-aware controls on a form.

For a list of all members of this type, see [TIBCDDataSource](#) members.

## Unit

[IBC](#)

## Syntax

```
TIBCDDataSource = class (TCRDataSource) ;
```

## Remarks

TIBCDDataSource provides an interface between an IBDAC dataset components and data-aware controls on a form.

TIBCDDataSource inherits its functionality directly from the TDataSource component. At design-time assign individual data-aware components' DataSource properties from their drop-down listboxes.

## Inheritance Hierarchy

[TCRDataSource](#)

**TIBCDDataSource**

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.9.1 Members

[TIBCDDataSource](#) class overview.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1(TIBCDbKeyField Class

A class representing the InterBase RDB\$DB KEY field.

For a list of all members of this type, see [TIBCDbKeyField](#) members.

## Unit

[IBC](#)

## Syntax

```
TIBCDbKeyField = class (TBytesField) ;
```

## Remarks

This class represents the InterBase RDB\$DB KEY field. It was implemented for the text view of DB KEY values and does not change TBytesField interface.

## See Also

- [Updating Data with IBDAC Dataset Components](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.10.1 Members

[TIBCDbKeyField](#) class overview.

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.1 TIBCEncryptor Class

The class that performs encrypting and decrypting of data.  
For a list of all members of this type, see [TIBCEncryptor](#) members.

**Unit**

[IBC](#)

**Syntax**

```
TIBCEncryptor = class(TCREncryptor);
```

**Inheritance Hierarchy**

[TCREncryptor](#)  
**TIBCEncryptor**

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.11.1 Members

[TIBCEncryptor](#) class overview.

**Properties**

Name	Description
<a href="#">DataHeader</a> (inherited from <a href="#">TCREncryptor</a> )	Specifies whether the additional information is stored with the encrypted data.
<a href="#">EncryptionAlgorithm</a> (inherited from <a href="#">TCREncryptor</a> )	Specifies the algorithm of data encryption.
<a href="#">HashAlgorithm</a> (inherited from <a href="#">TCREncryptor</a> )	Specifies the algorithm of generating hash data.
<a href="#">InvalidHashAction</a> (inherited from <a href="#">TCREncryptor</a> )	Specifies the action to perform on data fetching when hash data is invalid.
<a href="#">Password</a> (inherited from <a href="#">TCREncryptor</a> )	Used to set a password that is used to generate a key for encryption.

**Methods**

Name	Description
<a href="#">SetKey</a> (inherited from <a href="#">TCREncryptor</a> )	Sets a key, using which data is encrypted.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1 TIBCMetaData Class

A component for obtaining metainformation about database objects from the server. For a list of all members of this type, see [TIBCMetaData](#) members.

#### Unit

[IBC](#)

#### Syntax

```
TIBCMetaData = class (TDAMetaData) ;
```

#### Remarks

The TIBCMetaData component is used to obtain metainformation from the server about objects in the database, such as tables, table columns, stored procedures, etc.

#### Inheritance Hierarchy

[TMemDataSet](#)  
[TDAMetaData](#)  
**TIBCMetaData**

#### See Also

- [TCustomDADataset.Debug](#)
- [TCustomDASQL.Debug](#)
- A:Using DBMonitor

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.12.1 Members

[TIBCMetaData](#) class overview.

#### Properties

Name	Description
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">Connection</a> (inherited from <a href="#">TDAMetaData</a> )	Used to specify a connection object to use to connect to a data store.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">MetaDataKind</a> (inherited from <a href="#">TDAMetaData</a> )	Used to specify which kind of metainformation to show.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">Restrictions</a> (inherited from <a href="#">TDAMetaData</a> )	Used to provide one or more conditions restricting the list of objects to be described.
<a href="#">Transaction</a>	Used to determine the transaction under which queries to the server are executed.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.

## Methods

Name	Description
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">GetBlob</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">GetMetaDataKinds</a> (inherited from <a href="#">TDAMetaData</a> )	Used to get values acceptable in the MetaDataKind property.
<a href="#">GetRestrictions</a> (inherited from <a href="#">TDAMetaData</a> )	Used to find out which restrictions are applicable to a certain MetaDataKind.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.

<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">Prepare</a> (inherited from <a href="#">TMemDataSet</a> )	Allocates resources and creates field components for a dataset.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the <a href="#">ApplyUpdates</a> method while cached updates are enabled.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.

## Events

Name	Description
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.12.2 Properties

Properties of the **TIBCMetaData** class.  
For a complete list of the **TIBCMetaData** class members, see the [TIBCMetaData Members](#) topic.

## Public

Name	Description
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">Connection</a> (inherited from <a href="#">TDAMetaData</a> )	Used to specify a connection object to use to connect to a data store.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">GetBlob</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">GetMetaDataKinds</a> (inherited from <a href="#">TDAMetaData</a> )	Used to get values acceptable in the MetaDataKind property.
<a href="#">GetRestrictions</a> (inherited from <a href="#">TDAMetaData</a> )	Used to find out which restrictions are applicable to a certain MetaDataKind.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.

<a href="#"><u>MetaDataKind</u></a> (inherited from <a href="#"><u>TDAMetaData</u></a> )	Used to specify which kind of metainformation to show.
<a href="#"><u>OnUpdateError</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#"><u>OnUpdateRecord</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when a single update component can not handle the updates.
<a href="#"><u>Prepare</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Allocates resources and creates field components for a dataset.
<a href="#"><u>Prepared</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Determines whether a query is prepared for execution or not.
<a href="#"><u>RestoreUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Marks all records in the cache of updates as unapplied.
<a href="#"><u>Restrictions</u></a> (inherited from <a href="#"><u>TDAMetaData</u></a> )	Used to provide one or more conditions restricting the list of objects to be described.
<a href="#"><u>RevertRecord</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#"><u>SaveToXML</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#"><u>UnPrepare</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#"><u>UpdateRecordTypes</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#"><u>UpdateResult</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
<a href="#"><u>UpdatesPending</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to check the status of the cached updates buffer.
<a href="#"><u>UpdateStatus</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Indicates the current update status for the dataset when cached updates are enabled.

## Published

Name	Description
------	-------------



## [Transaction](#)

Used to determine the transaction under which queries to the server are executed.

### See Also

- [TIBCMetaData Class](#)
- [TIBCMetaData Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.12.2.1 Transaction Property

Used to determine the transaction under which queries to the server are executed.

### Class

[TIBCMetaData](#)

### Syntax

```
property Transaction: TIBCTransaction stored IsTransactionStored;
```

### Remarks

Use the Transaction property to determine the transaction under which queries to the server are executed.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1TIBCPParam Class

For a list of all members of this type, see [TIBCPParam](#) members.

### Unit

[IBC](#)

### Syntax

```
TIBCPParam = class(TDAParam);
```

### Example

### Inheritance Hierarchy

[TDAParam](#)

**TIBCPParam**

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.13.1 Members

[TIBCPParam](#) class overview.

## Properties

Name	Description
<a href="#">AsArray</a>	Used to specify the value of the parameter when it represents the value of the InterBase array type.
<a href="#">AsBlob</a> (inherited from <a href="#">TDAParam</a> )	Used to set and read the value of the BLOB parameter as string.
<a href="#">AsBlobRef</a> (inherited from <a href="#">TDAParam</a> )	Used to set and read the value of the BLOB parameter as a TBlob object.
<a href="#">AsFloat</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the value for a float field to a parameter.
<a href="#">AsIBlob</a>	Used to specify the value of the parameter when it represents the value of BLOB type.
<a href="#">AsInteger</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the value for an integer field to the parameter.
<a href="#">AsLargeInt</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the value for a LargeInteger field to the parameter.
<a href="#">AsMemo</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the value for a memo field to the parameter.
<a href="#">AsMemoRef</a> (inherited from <a href="#">TDAParam</a> )	Used to set and read the value of the memo parameter as a TBlob object.
<a href="#">AsSQLTimeStamp</a> (inherited from <a href="#">TDAParam</a> )	Used to specify the value of the parameter when it represents a SQL timestamp field.
<a href="#">AsString</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the string value to the parameter.
<a href="#">AsWideString</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the Unicode string value to the parameter.
<a href="#">DataType</a> (inherited from <a href="#">TDAParam</a> )	Indicates the data type of the parameter.
<a href="#">IsNull</a> (inherited from <a href="#">TDAParam</a> )	Used to indicate whether the value assigned to a parameter is NULL.
<a href="#">ParamType</a> (inherited from <a href="#">TDAParam</a> )	Used to indicate the type of use for a parameter.

[Size](#) (inherited from [TDAParam](#))

Specifies the size of a string type parameter.

[Value](#) (inherited from [TDAParam](#))

Used to represent the value of the parameter as Variant.

## Methods

Name	Description
<a href="#">AssignField</a> (inherited from <a href="#">TDAParam</a> )	Assigns field name and field value to a param.
<a href="#">AssignFieldValue</a> (inherited from <a href="#">TDAParam</a> )	Assigns the specified field properties and value to a parameter.
<a href="#">LoadFromFile</a> (inherited from <a href="#">TDAParam</a> )	Places the content of a specified file into a TDAParam object.
<a href="#">LoadFromStream</a> (inherited from <a href="#">TDAParam</a> )	Places the content from a stream into a TDAParam object.
<a href="#">SetBlobData</a>	Sets the parameter value from the memory buffer.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.13.2 Properties

Properties of the **TIBCPARAM** class.

For a complete list of the **TIBCPARAM** class members, see the [TIBCPARAM Members](#) topic.

## Public

Name	Description
<a href="#">AsArray</a>	Used to specify the value of the parameter when it represents the value of the InterBase array type.
<a href="#">AsBlob</a> (inherited from <a href="#">TDAParam</a> )	Used to set and read the value of the BLOB parameter as string.
<a href="#">AsBlobRef</a> (inherited from <a href="#">TDAParam</a> )	Used to set and read the value of the BLOB parameter as a TBlob object.
<a href="#">AsFloat</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the value for a float field to a parameter.
<a href="#">AsIBlob</a>	Used to specify the value of the parameter when it represents the value of BLOB type.

<a href="#">AsInteger</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the value for an integer field to the parameter.
<a href="#">AsLargeInt</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the value for a LargeInteger field to the parameter.
<a href="#">AsMemo</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the value for a memo field to the parameter.
<a href="#">AsMemoRef</a> (inherited from <a href="#">TDAParam</a> )	Used to set and read the value of the memo parameter as a TBlob object.
<a href="#">AssignField</a> (inherited from <a href="#">TDAParam</a> )	Assigns field name and field value to a param.
<a href="#">AssignFieldValue</a> (inherited from <a href="#">TDAParam</a> )	Assigns the specified field properties and value to a parameter.
<a href="#">AsSQLTimeStamp</a> (inherited from <a href="#">TDAParam</a> )	Used to specify the value of the parameter when it represents a SQL timestamp field.
<a href="#">AsString</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the string value to the parameter.
<a href="#">AsWideString</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the Unicode string value to the parameter.
<a href="#">IsNull</a> (inherited from <a href="#">TDAParam</a> )	Used to indicate whether the value assigned to a parameter is NULL.
<a href="#">LoadFromFile</a> (inherited from <a href="#">TDAParam</a> )	Places the content of a specified file into a TDAParam object.
<a href="#">LoadFromStream</a> (inherited from <a href="#">TDAParam</a> )	Places the content from a stream into a TDAParam object.
<a href="#">SetBlobData</a> (inherited from <a href="#">TDAParam</a> )	Overloaded. Writes the data from a specified buffer to BLOB.

## Published

Name	Description
<a href="#">DataType</a> (inherited from <a href="#">TDAParam</a> )	Indicates the data type of the parameter.
<a href="#">ParamType</a> (inherited from <a href="#">TDAParam</a> )	Used to indicate the type of use for a parameter.
<a href="#">Size</a> (inherited from <a href="#">TDAParam</a> )	Specifies the size of a string type parameter.
<a href="#">Value</a> (inherited from <a href="#">TDAParam</a> )	Used to represent the value of the parameter as Variant.

**See Also**

- [TIBCPParam Class](#)
- [TIBCPParam Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.13.2.1 AsArray Property

Used to specify the value of the parameter when it represents the value of the InterBase array type.

**Class**

[TIBCPParam](#)

**Syntax**

**property** AsArray: [TIBCArrary](#);

**Remarks**

Use the AsArray property to specify the value of the parameter when it represents the value of the InterBase array type.  
Setting AsArray will set the DataType property to ftArray.

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.13.2.2 AsIbBlob Property

Used to specify the value of the parameter when it represents the value of BLOB type.

**Class**

[TIBCPParam](#)

**Syntax**

**property** AsIbBlob: [TIBCBlob](#);

**Remarks**

Use the AsIbBlob property to specify the value of the parameter when it represents the value of BLOB type.  
Setting AsIBCBLOB will set the DataType property to ftIBCBLOB.

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.13.3 Methods

Methods of the **TIBCPParam** class.  
For a complete list of the **TIBCPParam** class members, see the [TIBCPParam Members](#) topic.

**Public**

Name	Description
------	-------------

<a href="#">AsBlob</a> (inherited from <a href="#">TDAParam</a> )	Used to set and read the value of the BLOB parameter as string.
<a href="#">AsBlobRef</a> (inherited from <a href="#">TDAParam</a> )	Used to set and read the value of the BLOB parameter as a TBlob object.
<a href="#">AsFloat</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the value for a float field to a parameter.
<a href="#">AsInteger</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the value for an integer field to the parameter.
<a href="#">AsLargeInt</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the value for a LargeInteger field to the parameter.
<a href="#">AsMemo</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the value for a memo field to the parameter.
<a href="#">AsMemoRef</a> (inherited from <a href="#">TDAParam</a> )	Used to set and read the value of the memo parameter as a TBlob object.
<a href="#">AssignField</a> (inherited from <a href="#">TDAParam</a> )	Assigns field name and field value to a param.
<a href="#">AssignFieldValue</a> (inherited from <a href="#">TDAParam</a> )	Assigns the specified field properties and value to a parameter.
<a href="#">AsSQLTimeStamp</a> (inherited from <a href="#">TDAParam</a> )	Used to specify the value of the parameter when it represents a SQL timestamp field.
<a href="#">AsString</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the string value to the parameter.
<a href="#">AsWideString</a> (inherited from <a href="#">TDAParam</a> )	Used to assign the Unicode string value to the parameter.
<a href="#">IsNull</a> (inherited from <a href="#">TDAParam</a> )	Used to indicate whether the value assigned to a parameter is NULL.
<a href="#">LoadFromFile</a> (inherited from <a href="#">TDAParam</a> )	Places the content of a specified file into a TDAParam object.
<a href="#">LoadFromStream</a> (inherited from <a href="#">TDAParam</a> )	Places the content from a stream into a TDAParam object.
<a href="#">SetBlobData</a>	Sets the parameter value from the memory buffer.

**Published**

Name	Description
------	-------------

[DataType](#) (inherited from [TDAParam](#))

Indicates the data type of the parameter.

[ParamType](#) (inherited from [TDAParam](#))

Used to indicate the type of use for a parameter.

[Size](#) (inherited from [TDAParam](#))

Specifies the size of a string type parameter.

[Value](#) (inherited from [TDAParam](#))

Used to represent the value of the parameter as Variant.

### See Also

- [TIBCPARAM Class](#)
- [TIBCPARAM Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.13.3.1 SetBlobData Method

Sets the parameter value from the memory buffer.

### Class

[TIBCPARAM](#)

### Syntax

```
procedure SetBlobData(Buffer: IntPtr; Size: integer);
```

#### Parameters

*Buffer*

Holds the pointer to the buffer with data.

*Size*

Holds the buffer size.

### Remarks

Call the SetBLOBData procedure to set the parameter value from the memory buffer. After this procedure call the DataType property is assigned to ftBLOB.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1 TIBCPARAMS Class

Used to control TIBCPARAM objects.

For a list of all members of this type, see [TIBCPARAMS](#) members.

### Unit

[IBC](#)

### Syntax

```
TIBCPARAMS = class (TDAPARAMS);
```

### Remarks

Use TIBCPARAMS to manage a list of TIBCPARAM objects for an object that uses field parameters. For example, TIBCStoredProc objects and TIBCQuery objects use

TIBCPParams objects to create and access their parameters.

## Inheritance Hierarchy

[TDAParams](#)

**TIBCPParams**

## See Also

- [TIBCPParam](#)
  - [TCustomDASQL.Params](#)
  - [TCustomDADataSet.Params](#)
  - [TIBCSQL.Params](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.14.1 Members

[TIBCPParams](#) class overview.

## Properties

Name	Description
<a href="#">Items</a>	Used to iterate through all field parameters.

## Methods

Name	Description
<a href="#">FindParam</a>	Searches a parameter with the name passed in Value.
<a href="#">ParamByName</a>	Searches a parameter with the name passed in Value.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.14.2 Properties

Properties of the **TIBCPParams** class.

For a complete list of the **TIBCPParams** class members, see the [TIBCPParams Members](#) topic.

## Public

Name	Description
<a href="#">FindParam</a> (inherited from <a href="#">TDAParams</a> )	Searches for a parameter with the specified name.
<a href="#">Items</a>	Used to iterate through all field parameters.
<a href="#">ParamByName</a> (inherited from <a href="#">TDAParams</a> )	Searches for a parameter with the specified name.

## See Also



- [TIBCParams Class](#)
- [TIBCParams Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.14.2.1 Items Property(Indexer)

Used to iterate through all field parameters.

### Class

[TIBCParams](#)

### Syntax

**property** Items[Index: integer]: [TIBCParam](#); **default;**  
**Parameters**

*Index*

Holds the index in the range 0..Count - 1.

### Remarks

Use the Items property to iterate through all field parameters. Index identifies the index in the range 0..Count - 1. Items can reference a particular parameter by its index, but the ParamByName method is preferred, so as to avoid depending on the order of the parameters.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.14.3 Methods

Methods of the **TIBCParams** class.

For a complete list of the **TIBCParams** class members, see the [TIBCParams Members](#) topic.

### Public

Name	Description
<a href="#">FindParam</a>	Searches a parameter with the name passed in Value.
<a href="#">Items</a> (inherited from <a href="#">TDAParams</a> )	Used to iterate through all parameters.
<a href="#">ParamByName</a>	Searches a parameter with the name passed in Value.

### See Also

- [TIBCParams Class](#)
- [TIBCParams Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.14.3.1 FindParam Method

Searches a parameter with the name passed in Value.

**Class**

[TIBCPParams](#)

**Syntax**

```
function FindParam(const Value: string): TIBCPParam;
```

**Parameters**

*Value*

Holds the parameter name.

**Return Value**

the parameter, if a match was found.

**Remarks**

Call the FindParam method to find a parameter with the name passed in Value. If a match is found, FindParam returns the parameter. Otherwise, it returns nil. Use this method rather than a direct reference to the Items property to avoid depending on the order of the entries.

To locate more than one parameter at a time by name, use the GetParamList method instead. To get only the value of a named parameter, use the ParamValues property.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.14.3.2 ParamByName Method

Searches a parameter with the name passed in Value.

**Class**

[TIBCPParams](#)

**Syntax**

```
function ParamByName(const Value: string): TIBCPParam;
```

**Parameters**

*Value*

Holds the parameter name.

**Return Value**

the parameter, if a match was found.

**Remarks**

Call the ParamByName method to find a parameter with the name passed in Value. If a match is found, ParamByName returns the parameter. Otherwise, an exception is raised. Use this method rather than a direct reference to the Items property to avoid depending on the order of the entries.

To locate a parameter by name without raising an exception if the parameter is not found, use the FindParam method.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1 TIBQuery Class

A component for executing queries and operating record sets. It also provides flexible way to update data.

For a list of all members of this type, see [TIBQuery](#) members.

#### Unit

[IBC](#)

#### Syntax

```
TIBQuery = class (TCustomIBQuery);
```

#### Remarks

TIBQuery is a direct descendant of the [TCustomIBDataSet](#) component. It publishes most of its inherited properties and events so that they can be manipulated at design-time.

Use TIBQuery to perform fetching, insertion, deletion and update of record by dynamically generated SQL statements. TIBQuery provides automatic blocking of records, their checking before edit and refreshing after post. Set SQL, SQLInsert, SQLDelete, SQLRefresh, and SQLUpdate properties to define SQL statements for subsequent accesses to the database server. There is no restriction to their syntax, so any SQL statement is allowed. Usually you need to use INSERT, DELETE, and UPDATE statements but you also may use stored procedures in more diverse cases. To modify records, you can specify KeyFields. If they are not specified, TIBQuery will retrieve primary keys for UpdatingTable from metadata. TIBQuery can automatically update only one table. Updating table is defined by the UpdatingTable property if this property is set. Otherwise, the table a field of which is the first field in the field list in the SELECT clause is used as an updating table.

The SQLInsert, SQLDelete, SQLUpdate, SQLRefresh properties support automatic binding of parameters which have identical names to fields captions. To retrieve the value of a field as it was before the operation use the field name with the 'OLD ' prefix. This is especially useful when doing field comparisons in the WHERE clause of the statement. Use the [TCustomDADataset.BeforeUpdateExecute](#) event to assign the value to additional parameters and the [TCustomDADataset.AfterUpdateExecute](#) event to read them.

#### Inheritance Hierarchy

[TMemDataSet](#)  
[TCustomDADataset](#)  
[TCustomIBDataSet](#)  
[TCustomIBQuery](#)  
**TIBQuery**

#### See Also

- [Updating Data with IBDAC Dataset Components](#)
- [Master/Detail Relationships](#)
- [TIBStoredProc](#)

- [TIBCTable](#)

© 1997-2013 Devart. All Rights Reserved.

17.13.1.15.1 Members

[TIBQuery](#) class overview.

## Properties

Name	Description
<a href="#">AutoCommit</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to automatically commit each update, insert or delete statement by database server.
<a href="#">BaseSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">Connection</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the connection in which the dataset will be executed.
<a href="#">Cursor</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
<a href="#">Debug</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#">DetailFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#">Disconnected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to keep dataset opened after connection is closed.
<a href="#">DMLRefresh</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to refresh record by the RETURNIG clause when insert is performed.
<a href="#">Encryption</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the options of the data encryption in a dataset.
<a href="#">FetchAll</a>	Defines whether to request all records of the query from database server when the dataset is being opened.

<a href="#">FetchRows</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to define the number of rows to be transferred across the network at the same time.
<a href="#">FilterSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#">FinalSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#">GeneratorMode</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify which method is used internally to generate a sequenced field.
<a href="#">GeneratorStep</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">IsQuery</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to check if the SQL statement returns rows.
<a href="#">KeyFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#">KeyGenerator</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.

<a href="#"><u>LockMode</u></a>	Used to specify what kind of lock will be performed when editing a record.
<a href="#"><u>MacroCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to get the number of macros associated with the Macros property.
<a href="#"><u>Macros</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Makes it possible to change SQL queries easily.
<a href="#"><u>MasterFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#"><u>MasterSource</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the data source component which binds current dataset to the master one.
<a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#"><u>ParamCheck</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#"><u>ParamCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate how many parameters are there in the Params property.
<a href="#"><u>Params</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#"><u>Plan</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get or set the PLAN clause of the SELECT statement.
<a href="#"><u>Prepared</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Determines whether a query is prepared for execution or not.
<a href="#"><u>ReadOnly</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#"><u>RefreshOptions</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate when the editing record is refreshed.
<a href="#"><u>RowsAffected</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

<a href="#">RowsDeleted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were deleted during the last query operation.
<a href="#">RowsFetched</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the number of the currently fetched rows.
<a href="#">RowsInserted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were inserted during the last query operation.
<a href="#">RowsUpdated</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were updated during the last query operation.
<a href="#">SQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to provide a SQL statement that a query component executes when its Open method is called.
<a href="#">SQLDelete</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.
<a href="#">SQLInsert</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#">SQLLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to perform a record lock.
<a href="#">SQLRefresh</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#">TCustomDADataset.RefreshRecord</a> procedure.
<a href="#">SQLType</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the typecode of the SQL statement being processed by the InterBase database server.
<a href="#">SQLUpdate</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#">Transaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine the transaction under which the query of this dataset executes.
<a href="#">UniDirectional</a> (inherited from <a href="#">TCustomDADataset</a> )	Used if an application does not need bidirectional access to records in the result set.

<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.
<a href="#">UpdateTransaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get or set the transaction for modifying a dataset.
<a href="#">UpdatingTable</a>	Used to specify which table in a query is assumed to be the target for subsequent data-modification queries as a result of user incentive to insert, update or delete records.

## Methods

Name	Description
<a href="#">AddWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">BreakExec</a> (inherited from <a href="#">TCustomDADataset</a> )	Breaks execution of the SQL statement on the server.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">CreateBlobStream</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#">CreateProcCall</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Assigns PL/SQL block that calls stored procedure to the SQL property.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">DeleteWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.



<a href="#"><u>Execute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Executes a SQL statement on the server.
<a href="#"><u>Executing</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether SQL statement is still being executed.
<a href="#"><u>Fetches</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to determine if TCustomDADataset has already fetched all rows.
<a href="#"><u>Fetching</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#"><u>FetchingAll</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#"><u>FindKey</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Searches for a record which contains specified field values.
<a href="#"><u>FindMacro</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether a specified macro exists in a dataset.
<a href="#"><u>FindNearest</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#"><u>FindParam</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Determines if a parameter with the specified name exists in a dataset.
<a href="#"><u>GetArray</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Retrieves a TIBCArrary object for a field when only its name is known.
<a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Retrieves a TIBCBlob object for a field when only its name is known.
<a href="#"><u>GetDataType</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#"><u>GetFieldObject</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Returns a multireference shared object from field.
<a href="#"><u>GetFieldPrecision</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves the precision of a number field.
<a href="#"><u>GetFieldScale</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves the scale of a number field.
<a href="#"><u>GetOrderBy</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves an ORDER BY clause from a SQL statement.

<a href="#">GotoCurrent</a> (inherited from <a href="#">TCustomDADataset</a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">Lock</a> (inherited from <a href="#">TCustomDADataset</a> )	Locks the current record.
<a href="#">MacroByName</a> (inherited from <a href="#">TCustomDADataset</a> )	Finds a Macro with the name passed in Name.
<a href="#">ParamByName</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Called to set or use parameter information for a specific parameter based on its name.
<a href="#">Prepare</a> (inherited from <a href="#">TCustomDADataset</a> )	Allocates, opens, and parses cursor for a query.
<a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )	Actuali es field values for the current record.
<a href="#">RestoreSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.

<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.

## Events

Name	Description
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">BeforeFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#">BeforeUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.15.2 Properties

Properties of the **TIBCQuery** class.  
For a complete list of the **TIBCQuery** class members, see the [TIBCQuery Members](#) topic.

## Public

Name	Description
------	-------------

<a href="#"><u>AddWhere</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#"><u>AfterExecute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs after a component has executed a query to database.
<a href="#"><u>AfterFetch</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs after dataset finishes fetching data from server.
<a href="#"><u>AfterUpdateExecute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#"><u>ApplyUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#"><u>AutoCommit</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to automatically commit each update, insert or delete statement by database server.
<a href="#"><u>BaseSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#"><u>BeforeFetch</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#"><u>BeforeUpdateExecute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#"><u>BreakExec</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Breaks execution of the SQL statement on the server.
<a href="#"><u>CachedUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#"><u>CancelUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#"><u>CommitUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Clears the cached updates buffer.
<a href="#"><u>Connection</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the connection in which the dataset will be executed.
<a href="#"><u>CreateBlobStream</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.

<a href="#">CreateProcCall</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Assigns PL/SQL block that calls stored procedure to the SQL property.
<a href="#">Cursor</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
<a href="#">Debug</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataset</a> )	Makes permanent changes to the database server.
<a href="#">DeleteWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#">DetailFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#">Disconnected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to keep dataset opened after connection is closed.
<a href="#">DMLRefresh</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to refresh record by the RETURNIG clause when insert is performed.
<a href="#">Encryption</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the options of the data encryption in a dataset.
<a href="#">Execute</a> (inherited from <a href="#">TCustomDADataset</a> )	Executes a SQL statement on the server.
<a href="#">Executing</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether SQL statement is still being executed.
<a href="#">Fetched</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine if TCustomDADataset has already fetched all rows.
<a href="#">Fetching</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#">FetchingAll</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#">FetchRows</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to define the number of rows to be transferred across the network at the same time.

<a href="#"><u>FilterSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#"><u>FinalSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#"><u>FindKey</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Searches for a record which contains specified field values.
<a href="#"><u>FindMacro</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether a specified macro exists in a dataset.
<a href="#"><u>FindNearest</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#"><u>FindParam</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Determines if a parameter with the specified name exists in a dataset.
<a href="#"><u>GeneratorMode</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify which method is used internally to generate a sequenced field.
<a href="#"><u>GeneratorStep</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
<a href="#"><u>GetArray</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Retrieves a TIBCArrary object for a field when only its name is known.
<a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Retrieves a TIBCBlob object for a field when only its name is known.
<a href="#"><u>GetDataType</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#"><u>GetFieldObject</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Returns a multireference shared object from field.
<a href="#"><u>GetFieldPrecision</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves the precision of a number field.
<a href="#"><u>GetFieldScale</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves the scale of a number field.
<a href="#"><u>GetOrderBy</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves an ORDER BY clause from a SQL statement.

<a href="#"><u>GotoCurrent</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#"><u>Handle</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#"><u>IndexFieldNames</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to check if the SQL statement returns rows.
<a href="#"><u>KeyFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#"><u>KeyGenerator</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#"><u>LocalConstraints</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#"><u>LocalUpdate</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to prevent implicit update of rows on database server.
<a href="#"><u>Locate</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#"><u>LocateEx</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Excludes features that don't need to be included to the <a href="#"><u>TMemDataSet.Locate</u></a> method of TDataSet.
<a href="#"><u>Lock</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Locks the current record.
<a href="#"><u>MacroByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Finds a Macro with the name passed in Name.
<a href="#"><u>MacroCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to get the number of macros associated with the Macros property.
<a href="#"><u>Macros</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Makes it possible to change SQL queries easily.

<a href="#"><u>MasterFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#"><u>MasterSource</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the data source component which binds current dataset to the master one.
<a href="#"><u>OnUpdateError</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#"><u>OnUpdateRecord</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when a single update component can not handle the updates.
<a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Called to set or use parameter information for a specific parameter based on its name.
<a href="#"><u>ParamCheck</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#"><u>ParamCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate how many parameters are there in the Params property.
<a href="#"><u>Params</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#"><u>Plan</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get or set the PLAN clause of the SELECT statement.
<a href="#"><u>Prepare</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Allocates, opens, and parses cursor for a query.
<a href="#"><u>Prepared</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Determines whether a query is prepared for execution or not.
<a href="#"><u>ReadOnly</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#"><u>RefreshOptions</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate when the editing record is refreshed.



<a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )	Actualizes field values for the current record.
<a href="#">RestoreSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchronizes the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">RowsAffected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#">RowsDeleted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were deleted during the last query operation.
<a href="#">RowsFetched</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the number of the currently fetched rows.
<a href="#">RowsInserted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were inserted during the last query operation.
<a href="#">RowsUpdated</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were updated during the last query operation.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to provide a SQL statement that a query component executes when its Open method is called.

<a href="#">SQLDelete</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.
<a href="#">SQLInsert</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#">SQLLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to perform a record lock.
<a href="#">SQLRefresh</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#">TCustomDADataset.RefreshRecord</a> procedure.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">SQLType</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the typecode of the SQL statement being processed by the InterBase database server.
<a href="#">SQLUpdate</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#">Transaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine the transaction under which the query of this dataset executes.
<a href="#">UniDirectional</a> (inherited from <a href="#">TCustomDADataset</a> )	Used if an application does not need bidirectional access to records in the result set.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the <a href="#">ApplyUpdates</a> method while cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Indicates the current update status for the dataset when cached updates are enabled.

[UpdateTransaction](#) (inherited from [TCustomIBCDataset](#))

Used to get or set the transaction for modifying a dataset.

## Published

Name	Description
<a href="#">FetchAll</a>	Defines whether to request all records of the query from database server when the dataset is being opened.
<a href="#">LockMode</a>	Used to specify what kind of lock will be performed when editing a record.
<a href="#">UpdatingTable</a>	Used to specify which table in a query is assumed to be the target for subsequent data-modification queries as a result of user incentive to insert, update or delete records.

## See Also

- [TIBCQuery Class](#)
- [TIBCQuery Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.15.2.1 FetchAll Property

Defines whether to request all records of the query from database server when the dataset is being opened.

## Class

[TIBCQuery](#)

## Syntax

```
property FetchAll: boolean;
```

## Remarks

When set to True, all records of the query are requested from database server when the dataset is being opened. When set to False, records are retrieved when a data-aware component or a program requests it. If a query can return a lot of records, set this property to False if initial response time is important.

When the FetchAll property is False, the first call to [TMemDataSet.Locate](#) and [TMemDataSet.LocateEx](#) methods may take a lot of time to retrieve additional records to the client side.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.15.2.2 LockMode Property

Used to specify what kind of lock will be performed when editing a record.

### Class

[TIBCQuery](#)

### Syntax

```
property LockMode: TLockMode default lmNone;
```

### Remarks

Use the LockMode property to define what kind of lock will be performed when editing a record. Locking a record is useful in creating multi-user applications. It prevents modification of a record by several users at the same time. Locking is performed by the RefreshRecord method. The default value is lmNone.

### See Also

- [TIBCStoredProc.LockMode](#)
  - [TIBCTable.LockMode](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.15.2.3 UpdatingTable Property

Used to specify which table in a query is assumed to be the target for subsequent data-modification queries as a result of user incentive to insert, update or delete records.

### Class

[TIBCQuery](#)

### Syntax

```
property UpdatingTable: string;
```

### Remarks

Use the UpdatingTable property to specify which table in a query is assumed to be the target for the subsequent data-modification queries as a result of user incentive to insert, update or delete records.

This property is used on Insert, Update, Delete or RefreshRecord (see also [TCustomIBCDataset.Options](#)) if appropriate SQL (SQLInsert, SQLUpdate or SQLDelete) is not provided.

If UpdatingTable is not set then the first table used in a query is assumed to be the target.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1 TIBCSQL Class

A component for executing SQL statements and calling stored procedures on the database server.

For a list of all members of this type, see [TIBCSQL](#) members.

#### Unit

[IBC](#)

#### Syntax

```
TIBCSQL = class (TCustomDASQL) ;
```

#### Remarks

The TIBCSQL component is a direct descendant of the [TCustomDASQL](#) class.

Use The TIBCSQL component when a client application must execute SQL statement or the PL/SQL block, and call stored procedure on the database server. The SQL statement should not retrieve rows from the database.

#### Inheritance Hierarchy

[TCustomDASQL](#)  
**TIBCSQL**

#### See Also

- [TIBQuery](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.16.1 Members

[TIBCSQL](#) class overview.

#### Properties

Name	Description
<a href="#">ChangeCursor</a> (inherited from <a href="#">TCustomDASQL</a> )	Enables or disables changing screen cursor when executing commands in the NonBlocking mode.
<a href="#">Connection</a>	Used to set or return the connection associated with the query.
<a href="#">Debug</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to display executing statement, all its parameters' values, and the type of parameters.

[DescribeParams](#)

Used to specify whether to query [TIBCPParam](#) properties from the server when executing the [TCustomDASQL.Prepare](#) method.

[FinalSQL](#) (inherited from [TCustomDASQL](#))

Used to return a SQL statement with expanded macros.

[Handle](#)

Used to specify the handle for the SQL statement of TIBCSQL.

[MacroCount](#) (inherited from [TCustomDASQL](#))

Used to get the number of macros associated with the Macros property.

[Macros](#) (inherited from [TCustomDASQL](#))

Makes it possible to change SQL queries easily.

[ParamCheck](#) (inherited from [TCustomDASQL](#))

Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.

[ParamCount](#) (inherited from [TCustomDASQL](#))

Indicates the number of parameters in the Params property.

[Params](#)

Contains the parameters for SQL statement.

[ParamValues](#) (inherited from [TCustomDASQL](#))

Used to get or set the values of individual field parameters that are identified by name.

[Prepared](#) (inherited from [TCustomDASQL](#))

Used to indicate whether a query is prepared for execution.

[RowsAffected](#) (inherited from [TCustomDASQL](#))

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

[SQL](#) (inherited from [TCustomDASQL](#))

Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.

[SQLType](#)

Used to provide the typecode of the SQL statement being processed by the InterBase server.

[Transaction](#)

Used to set or return the transaction to be used by the component.

## Methods

Name	Description
<a href="#">BreakExec</a>	Breaks execution of a SQL statement on the server.
<a href="#">CreateProcCall</a>	Assigns SQL block that calls stored procedure specified by Name to SQL property.
<a href="#">Execute</a> (inherited from <a href="#">TCustomDASQL</a> )	Overloaded. Executes SQL commands.
<a href="#">ExecuteNext</a>	Provides data that is returned by SQL statement through the out parameters.
<a href="#">Executing</a> (inherited from <a href="#">TCustomDASQL</a> )	Checks whether TCustomDASQL still executes a SQL statement.
<a href="#">FindMacro</a> (inherited from <a href="#">TCustomDASQL</a> )	Searches for a macro with the specified name.
<a href="#">FindParam</a>	Searches a parameter with the specified name.
<a href="#">MacroByName</a> (inherited from <a href="#">TCustomDASQL</a> )	Finds a Macro with the name passed in Name.
<a href="#">ParamByName</a>	Searches a parameter with the specified name.
<a href="#">Prepare</a> (inherited from <a href="#">TCustomDASQL</a> )	Allocates, opens, and parses cursor for a query.
<a href="#">UnPrepare</a> (inherited from <a href="#">TCustomDASQL</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">WaitExecuting</a> (inherited from <a href="#">TCustomDASQL</a> )	Waits until TCustomDASQL executes a SQL statement.

## Events

Name	Description
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDASQL</a> )	Occurs after a SQL statement has been executed.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.16.2 Properties

Properties of the **TIBCSQL** class.  
For a complete list of the **TIBCSQL** class members, see the [TIBCSQL Members](#) topic.

## Public

Name	Description
------	-------------

<a href="#"><u>AfterExecute</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Occurs after a SQL statement has been executed.
<a href="#"><u>ChangeCursor</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Enables or disables changing screen cursor when executing commands in the NonBlocking mode.
<a href="#"><u>Debug</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#"><u>Execute</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Overloaded. Executes SQL commands.
<a href="#"><u>Executing</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Checks whether TCustomDASQL still executes a SQL statement.
<a href="#"><u>FinalSQL</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Used to return a SQL statement with expanded macros.
<a href="#"><u>FindMacro</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Searches for a macro with the specified name.
<a href="#"><u>FindParam</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Finds a parameter with the specified name.
<a href="#"><u>Handle</u></a>	Used to specify the handle for the SQL statement of TIBCSQL.
<a href="#"><u>MacroByName</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Finds a Macro with the name passed in Name.
<a href="#"><u>MacroCount</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Used to get the number of macros associated with the Macros property.
<a href="#"><u>Macros</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Makes it possible to change SQL queries easily.
<a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Finds a parameter with the specified name.
<a href="#"><u>ParamCheck</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.
<a href="#"><u>ParamCount</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Indicates the number of parameters in the Params property.
<a href="#"><u>ParamValues</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Used to get or set the values of individual field parameters that are identified by name.
<a href="#"><u>Prepare</u></a> (inherited from <a href="#"><u>TCustomDASQL</u></a> )	Allocates, opens, and parses cursor for a query.



<a href="#">Prepared</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to indicate whether a query is prepared for execution.
<a href="#">RowsAffected</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#">SQL</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.
<a href="#">SQLType</a>	Used to provide the typecode of the SQL statement being processed by the InterBase server.
<a href="#">UnPrepare</a> (inherited from <a href="#">TCustomDASQL</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">WaitExecuting</a> (inherited from <a href="#">TCustomDASQL</a> )	Waits until TCustomDASQL executes a SQL statement.

## Published

Name	Description
<a href="#">Connection</a>	Used to set or return the connection associated with the query.
<a href="#">DescribeParams</a>	Used to specify whether to query <a href="#">TIBCPParam</a> properties from the server when executing the <a href="#">TCustomDASQL.Prepare</a> method.
<a href="#">Params</a>	Contains the parameters for SQL statement.
<a href="#">Transaction</a>	Used to set or return the transaction to be used by the component.

## See Also

- [TIBCSQL Class](#)
- [TIBCSQL Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

17.13.1.16.2.1 Connection Property

Used to set or return the connection associated with the query.

## Class

[TIBCSQL](#)**Syntax**

**property** Connection: [TIBConnection](#);

**Remarks**

Use the Connection property to set or return the connection associated with the query.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.16.2.2 DescribeParams Property

Used to specify whether to query [TIBCPParam](#) properties from the server when executing the [TCustomDASQL.Prepare](#) method.

**Class**[TIBCSQL](#)**Syntax**

**property** DescribeParams: boolean **default** False;

**Remarks**

Specifies whether to query [TIBCPParam](#) properties (Name, ParamType, DataType, Size, TableName) from the server when executing the [TCustomDASQL.Prepare](#) method. The default value is False.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.16.2.3 Handle Property

Used to specify the handle for the SQL statement of TIBCSQL.

**Class**[TIBCSQL](#)**Syntax**

**property** Handle: TISC\_STMT\_HANDLE;

**Remarks**

Use the Handle property to specify the handle for the SQL statement of TIBCSQL.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.16.2.4 Params Property

Contains the parameters for SQL statement.

**Class**[TIBCSQL](#)**Syntax**

```
property Params: TIBCPParams stored False;
```

### Remarks

The Params property is used to hold the parameters for SQL statement. Access Params at runtime to view and set parameter names, values, and data types dynamically (at design time use the Parameters editor to set parameter information). Params is a zero-based array of parameter records. Index specifies the array element to access.

An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName.

### Example

Setting parameters in runtime.

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
  with IBCSQL do  
  begin  
    SQL.Clear;  
    SQL.Add('INSERT INTO Temp Table(Id, Name)');  
    SQL.Add('VALUES (:id, :Name)');  
    ParamByName('Id').AsInteger := 55;  
    Params[1].AsString := ' Green';  
    Execute;  
  end;  
end;
```

### See Also

- [TIBCPParam](#)
- [FindParam](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.16.2.5 SQLType Property

Used to provide the typecode of the SQL statement being processed by the InterBase server.

### Class

[TIBCSQL](#)

### Syntax

```
property SQLType: integer;
```

### Remarks

Use the SQLType property to get the typecode of the SQL statement being processed by the InterBase server.

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.16.2.6 Transaction Property

Used to set or return the transaction to be used by the component.

**Class**

[TIBCSQL](#)

**Syntax**

**property** Transaction: [TIBCTransaction](#) **stored** IsTransactionStored;

**Remarks**

Use the Transaction property to set or return the transaction to be used by the component.

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.16.3 Methods

Methods of the **TIBCSQL** class.

For a complete list of the **TIBCSQL** class members, see the [TIBCSQL Members](#) topic.

**Public**

Name	Description
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDASQL</a> )	Occurs after a SQL statement has been executed.
<a href="#">BreakExec</a>	Breaks execution of a SQL statement on the server.
<a href="#">ChangeCursor</a> (inherited from <a href="#">TCustomDASQL</a> )	Enables or disables changing screen cursor when executing commands in the NonBlocking mode.
<a href="#">Connection</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to specify a connection object to use to connect to a data store.
<a href="#">CreateProcCall</a>	Assigns SQL block that calls stored procedure specified by Name to SQL property.
<a href="#">Debug</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#">Execute</a> (inherited from <a href="#">TCustomDASQL</a> )	Overloaded. Executes SQL commands.
<a href="#">ExecuteNext</a>	Provides data that is returned by SQL statement through the out parameters.
<a href="#">Executing</a> (inherited from <a href="#">TCustomDASQL</a> )	Checks whether TCustomDASQL still executes a SQL statement.

<a href="#">FinalSQL</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to return a SQL statement with expanded macros.
<a href="#">FindMacro</a> (inherited from <a href="#">TCustomDASQL</a> )	Searches for a macro with the specified name.
<a href="#">FindParam</a>	Searches a parameter with the specified name.
<a href="#">MacroByName</a> (inherited from <a href="#">TCustomDASQL</a> )	Finds a Macro with the name passed in Name.
<a href="#">MacroCount</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to get the number of macros associated with the Macros property.
<a href="#">Macros</a> (inherited from <a href="#">TCustomDASQL</a> )	Makes it possible to change SQL queries easily.
<a href="#">ParamByName</a>	Searches a parameter with the specified name.
<a href="#">ParamCheck</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.
<a href="#">ParamCount</a> (inherited from <a href="#">TCustomDASQL</a> )	Indicates the number of parameters in the Params property.
<a href="#">Params</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to contain parameters for a SQL statement.
<a href="#">ParamValues</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to get or set the values of individual field parameters that are identified by name.
<a href="#">Prepare</a> (inherited from <a href="#">TCustomDASQL</a> )	Allocates, opens, and parses cursor for a query.
<a href="#">Prepared</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to indicate whether a query is prepared for execution.
<a href="#">RowsAffected</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#">SQL</a> (inherited from <a href="#">TCustomDASQL</a> )	Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.
<a href="#">UnPrepare</a> (inherited from <a href="#">TCustomDASQL</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.

[WaitExecuting](#) (inherited from [TCustomDASQL](#)) Waits until TCustomDASQL executes a SQL statement.

**See Also**

- [TIBCSQL Class](#)
  - [TIBCSQL Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.16.3.1 BreakExec Method

Breaks execution of a SQL statement on the server.

**Class**

[TIBCSQL](#)

**Syntax**

```
procedure BreakExec;
```

**Remarks**

Call the BreakExec method to break execution of the SQL statement on the server. It makes sense to call BreakExec only from another thread.

**See Also**

- [TCustomDASQL.Execute](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.16.3.2 CreateProcCall Method

Assigns SQL block that calls stored procedure specified by Name to SQL property.

**Class**

[TIBCSQL](#)

**Syntax**

```
procedure CreateProcCall(const Name: string);
```

**Parameters**

*Name*

Holds the stored procedure name.

**Remarks**

Call CreateProcCall to assign SQL block that calls stored procedure specified by Name to SQL property. Retrieves the information about parameters of the procedure from InterBase. After calling CreateProcCall you can execute stored procedure by Execute method.

## See Also

- [TCustomDASQL.Execute](#)
- [TCustomDAConnection.ExecProc](#)
- [TIBCSStoredProc](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.16.3.3 ExecuteNext Method

Provides data that is returned by SQL statement through the out parameters.

## Class

[TIBCSQL](#)

## Syntax

```
function ExecuteNext: boolean;
```

### Return Value

True, when the data were read and False when it reaches the end of the dataset.

## Remarks

Call the ExecuteNext method to get data that is returned by SQL statement through the out parameters. That can be select statement or execution of stored procedure that returns dataset. ExecuteNext returns True when the data were read and False when it reaches the end of dataset.

## Example

In the example below all the data is read from the Department column of the Department table in the InterBase sample database Employee.gdb.

```
IBCSql.Text := 'select DEPARTMENT from DEPT';  
with TParam(IBCSql.Params.Add) do begin  
    ParamType := ptOutPut;  
    DataType := ftInteger;  
    Name := 'DEPARTMENT';  
end;  
IBCSql.Prepare; //To avoid auto unprepare after Execute  
IBCSql.Execute;  
while IBCSql.ExecuteNext do begin  
    Memo.Lines.Add(IBCSql.ParamByName('DEPARTMENT').AsString);  
end;  
IBCSql.UnPrepare;
```

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.16.3.4 FindParam Method

Searches a parameter with the specified name.

## Class

[TIBCSQL](#)

### Syntax

```
function FindParam(const Value: string): TIBCPParam;
```

#### Parameters

*Value*

Holds the parameter name.

#### Return Value

the parameter, if a match was found.

### Remarks

Call the FindParam method to find a parameter with the name passed in the Name argument. If a match is found, FindParam returns the parameter. Otherwise, it returns nil.

### See Also

- [TIBCPParam](#)
  - [ParamByName](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.16.3.5 ParamByName Method

Searches a parameter with the specified name.

### Class

[TIBCSQL](#)

### Syntax

```
function ParamByName(const Value: string): TIBCPParam;
```

#### Parameters

*Value*

Holds the parameter name.

#### Return Value

the parameter, if a match was found.

### Remarks

Call the ParamByName method to find a parameter with the name passed in Name argument.  
If a match is found, ParamByName returns the parameter. Otherwise, an exception is raised.

### Example

```
TIBCSQL1.Execute;
```



```
Edit1.Text := IBCSQL1.ParamsByName('Contact').AsString;
```

## See Also

- [TIBCPParam](#)
- [FindParam](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1 TIBCSStoredProc Class

A component for accessing and executing stored procedures and functions.  
For a list of all members of this type, see [TIBCSStoredProc](#) members.

## Unit

[IBC](#)

## Syntax

```
TIBCSStoredProc = class(TCustomIBCQuery);
```

## Remarks

Use TIBCSStoredProc to access stored procedures on the database server.  
You need only to define the StoredProcName property, and the SQL statement to call the stored procedure will be generated automatically.  
Use the Execute method at runtime to generate request that instructs server to execute procedure and PrepareSQL to describe parameters at run time

## Inheritance Hierarchy

[TMemDataSet](#)  
[TCustomDADataset](#)  
[TCustomIBCDataset](#)  
[TCustomIBCQuery](#)  
**TIBCSStoredProc**

## See Also

- [TIBCQuery](#)
- [TIBCSQL](#)
- [Updating Data with IBDAC Dataset Components](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.17.1 Members

[TIBCSStoredProc](#) class overview.

## Properties

Name	Description
------	-------------

<a href="#"><u>AutoCommit</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to automatically commit each update, insert or delete statement by database server.
<a href="#"><u>BaseSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#"><u>CachedUpdates</u></a> (inherited from <a href="#"><u>TMemDataset</u></a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#"><u>Connection</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the connection in which the dataset will be executed.
<a href="#"><u>Cursor</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
<a href="#"><u>Debug</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#"><u>DetailFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#"><u>Disconnected</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to keep dataset opened after connection is closed.
<a href="#"><u>DMLRefresh</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to refresh record by the RETURNIG clause when insert is performed.
<a href="#"><u>Encryption</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the options of the data encryption in a dataset.
<a href="#"><u>FetchRows</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to define the number of rows to be transferred across the network at the same time.
<a href="#"><u>FilterSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#"><u>FinalSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.

<a href="#">GeneratorMode</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify which method is used internally to generate a sequenced field.
<a href="#">GeneratorStep</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">IsQuery</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to check if the SQL statement returns rows.
<a href="#">KeyFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#">KeyGenerator</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">LockMode</a>	Used to specify what kind of lock will be performed when editing a record.
<a href="#">MacroCount</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to get the number of macros associated with the Macros property.
<a href="#">Macros</a> (inherited from <a href="#">TCustomDADataset</a> )	Makes it possible to change SQL queries easily.

<a href="#"><u>MasterFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#"><u>MasterSource</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the data source component which binds current dataset to the master one.
<a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#"><u>ParamCheck</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#"><u>ParamCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate how many parameters are there in the Params property.
<a href="#"><u>Params</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#"><u>Plan</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get or set the PLAN clause of the SELECT statement.
<a href="#"><u>Prepared</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Determines whether a query is prepared for execution or not.
<a href="#"><u>ReadOnly</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#"><u>RefreshOptions</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate when the editing record is refreshed.
<a href="#"><u>RowsAffected</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#"><u>RowsDeleted</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to indicate the number of rows that were deleted during the last query operation.
<a href="#"><u>RowsFetched</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get the number of the currently fetched rows.
<a href="#"><u>RowsInserted</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to indicate the number of rows that were inserted during the last query operation.

<a href="#">RowsUpdated</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were updated during the last query operation.
<a href="#">SQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to provide a SQL statement that a query component executes when its Open method is called.
<a href="#">SQLDelete</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.
<a href="#">SQLInsert</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#">SQLLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to perform a record lock.
<a href="#">SQLRefresh</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#">TCustomDADataset.RefreshRecord</a> procedure.
<a href="#">SQLType</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the typecode of the SQL statement being processed by the InterBase database server.
<a href="#">SQLUpdate</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#">StoredProcName</a>	Used to specify the name of the stored procedure to call on the server.
<a href="#">Transaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine the transaction under which the query of this dataset executes.
<a href="#">UniDirectional</a> (inherited from <a href="#">TCustomDADataset</a> )	Used if an application does not need bidirectional access to records in the result set.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.

[UpdateTransaction](#) (inherited from [TCustomIBCDataset](#))

Used to get or set the transaction for modifying a dataset.

## Methods

Name	Description
<a href="#">AddWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">BreakExec</a> (inherited from <a href="#">TCustomDADataset</a> )	Breaks execution of the SQL statement on the server.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">CreateBlobStream</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#">CreateProcCall</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Assigns PL/SQL block that calls stored procedure to the SQL property.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">DeleteWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#">ExecProc</a>	Executes a SQL statement on the server.
<a href="#">Execute</a> (inherited from <a href="#">TCustomDADataset</a> )	Executes a SQL statement on the server.
<a href="#">Executing</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether SQL statement is still being executed.
<a href="#">Fetched</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine if TCustomDADataset has already fetched all rows.
<a href="#">Fetching</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is still fetching rows.

<a href="#">FetchingAll</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#">FindKey</a> (inherited from <a href="#">TCustomDADataset</a> )	Searches for a record which contains specified field values.
<a href="#">FindMacro</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether a specified macro exists in a dataset.
<a href="#">FindNearest</a> (inherited from <a href="#">TCustomDADataset</a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#">FindParam</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Determines if a parameter with the specified name exists in a dataset.
<a href="#">GetArray</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Retrieves a TIBCArrary object for a field when only its name is known.
<a href="#">GetBlob</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Retrieves a TIBCBlob object for a field when only its name is known.
<a href="#">GetDataType</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#">GetFieldObject</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns a multireference shared object from field.
<a href="#">GetFieldPrecision</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the precision of a number field.
<a href="#">GetFieldScale</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the scale of a number field.
<a href="#">GetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves an ORDER BY clause from a SQL statement.
<a href="#">GotoCurrent</a> (inherited from <a href="#">TCustomDADataset</a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.

<a href="#">Lock</a> (inherited from <a href="#">TCustomDADataset</a> )	Locks the current record.
<a href="#">MacroByName</a> (inherited from <a href="#">TCustomDADataset</a> )	Finds a Macro with the name passed in Name.
<a href="#">ParamByName</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Called to set or use parameter information for a specific parameter based on its name.
<a href="#">Prepare</a>	Describes the stored procedure parameters.
<a href="#">PrepareSQL</a>	Describes the stored procedure parameters.
<a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )	Actuali es field values for the current record.
<a href="#">RestoreSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.



[UpdateStatus](#) (inherited from [TMemDataSet](#))

Indicates the current update status for the dataset when cached updates are enabled.

## Events

Name	Description
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">BeforeFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#">BeforeUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.17.2 Properties

Properties of the **TIBCStoredProc** class.

For a complete list of the **TIBCStoredProc** class members, see the [TIBCStoredProc Members](#) topic.

## Public

Name	Description
<a href="#">AddWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after executing insert, delete, update, lock and refresh operations.

<a href="#"><u>ApplyUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#"><u>AutoCommit</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to automatically commit each update, insert or delete statement by database server.
<a href="#"><u>BaseSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#"><u>BeforeFetch</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#"><u>BeforeUpdateExecute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#"><u>BreakExec</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Breaks execution of the SQL statement on the server.
<a href="#"><u>CachedUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#"><u>CancelUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#"><u>CommitUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Clears the cached updates buffer.
<a href="#"><u>Connection</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the connection in which the dataset will be executed.
<a href="#"><u>CreateBlobStream</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#"><u>CreateProcCall</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Assigns PL/SQL block that calls stored procedure to the SQL property.
<a href="#"><u>Cursor</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
<a href="#"><u>Debug</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to display executing statement, all its parameters' values, and the type of parameters.

<a href="#"><u>DeferredPost</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Makes permanent changes to the database server.
<a href="#"><u>DeleteWhere</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#"><u>DetailFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#"><u>Disconnected</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to keep dataset opened after connection is closed.
<a href="#"><u>DMLRefresh</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to refresh record by the RETURNIG clause when insert is performed.
<a href="#"><u>Encryption</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the options of the data encryption in a dataset.
<a href="#"><u>Execute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Executes a SQL statement on the server.
<a href="#"><u>Executing</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether SQL statement is still being executed.
<a href="#"><u>Fetches</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to determine if TCustomDADataset has already fetched all rows.
<a href="#"><u>Fetching</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#"><u>FetchingAll</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#"><u>FetchRows</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to define the number of rows to be transferred across the network at the same time.
<a href="#"><u>FilterSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#"><u>FinalSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#"><u>FindKey</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Searches for a record which contains specified field values.

<a href="#">FindMacro</a> (inherited from <a href="#">TCustomDADataset</a> )	Indicates whether a specified macro exists in a dataset.
<a href="#">FindNearest</a> (inherited from <a href="#">TCustomDADataset</a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#">FindParam</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Determines if a parameter with the specified name exists in a dataset.
<a href="#">GeneratorMode</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify which method is used internally to generate a sequenced field.
<a href="#">GeneratorStep</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
<a href="#">GetArray</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Retrieves a TIBCArrary object for a field when only its name is known.
<a href="#">GetBlob</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Retrieves a TIBCBlob object for a field when only its name is known.
<a href="#">GetDataTypes</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#">GetFieldObject</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns a multireference shared object from field.
<a href="#">GetFieldPrecision</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the precision of a number field.
<a href="#">GetFieldScale</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the scale of a number field.
<a href="#">GetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves an ORDER BY clause from a SQL statement.
<a href="#">GotoCurrent</a> (inherited from <a href="#">TCustomDADataset</a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.

<a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to check if the SQL statement returns rows.
<a href="#"><u>KeyFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#"><u>KeyGenerator</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#"><u>LocalConstraints</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#"><u>LocalUpdate</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to prevent implicit update of rows on database server.
<a href="#"><u>Locate</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#"><u>LocateEx</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Excludes features that don't need to be included to the <a href="#"><u>TMemDataSet.Locate</u></a> method of TDataSet.
<a href="#"><u>Lock</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Locks the current record.
<a href="#"><u>MacroByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Finds a Macro with the name passed in Name.
<a href="#"><u>MacroCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to get the number of macros associated with the Macros property.
<a href="#"><u>Macros</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Makes it possible to change SQL queries easily.
<a href="#"><u>MasterFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#"><u>MasterSource</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the data source component which binds current dataset to the master one.

<a href="#"><u>OnUpdateError</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#"><u>OnUpdateRecord</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when a single update component can not handle the updates.
<a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Called to set or use parameter information for a specific parameter based on its name.
<a href="#"><u>ParamCheck</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#"><u>ParamCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate how many parameters are there in the Params property.
<a href="#"><u>Params</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#"><u>Plan</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get or set the PLAN clause of the SELECT statement.
<a href="#"><u>Prepare</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Allocates, opens, and parses cursor for a query.
<a href="#"><u>Prepared</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Determines whether a query is prepared for execution or not.
<a href="#"><u>ReadOnly</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#"><u>RefreshOptions</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate when the editing record is refreshed.
<a href="#"><u>RefreshRecord</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Actuali es field values for the current record.
<a href="#"><u>RestoreSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#"><u>RestoreUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Marks all records in the cache of updates as unapplied.

<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchronizes the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">RowsAffected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#">RowsDeleted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were deleted during the last query operation.
<a href="#">RowsFetched</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the number of the currently fetched rows.
<a href="#">RowsInserted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were inserted during the last query operation.
<a href="#">RowsUpdated</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were updated during the last query operation.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to provide a SQL statement that a query component executes when its Open method is called.
<a href="#">SQLDelete</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.
<a href="#">SQLInsert</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#">SQLLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to perform a record lock.

<a href="#">SQLRefresh</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#">TCustomDADataset.RefreshRecord</a> procedure.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">SQLType</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the typecode of the SQL statement being processed by the InterBase database server.
<a href="#">SQLUpdate</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#">Transaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine the transaction under which the query of this dataset executes.
<a href="#">UniDirectional</a> (inherited from <a href="#">TCustomDADataset</a> )	Used if an application does not need bidirectional access to records in the result set.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the <a href="#">ApplyUpdates</a> method while cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.
<a href="#">UpdateTransaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get or set the transaction for modifying a dataset.

**Published**

Name	Description
------	-------------



[LockMode](#)

Used to specify what kind of lock will be performed when editing a record.

[StoredProcName](#)

Used to specify the name of the stored procedure to call on the server.

**See Also**

- [TIBCStoredProc Class](#)
- [TIBCStoredProc Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.17.2.1 LockMode Property

Used to specify what kind of lock will be performed when editing a record.

**Class**[TIBCStoredProc](#)**Syntax**

```
property LockMode: TLockMode default lmNone;
```

**Remarks**

Use the LockMode property to define what kind of lock will be performed when editing a record. Locking a record is useful in creating multi-user applications. It prevents modification of a record by several users at the same time. Locking is performed by the RefreshRecord method. The default value is lmNone.

**See Also**

- [TIBCQuery.LockMode](#)
- [TIBCTable.LockMode](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.17.2.2 StoredProcName Property

Used to specify the name of the stored procedure to call on the server.

**Class**[TIBCStoredProc](#)**Syntax**

```
property StoredProcName: string;
```

**Remarks**

Use the StoredProcName property to specify the name of the stored procedure to call on the server. If StoredProcName does not match the name of an existing

stored procedure on the server, then when the application attempts to prepare the procedure prior to execution, an exception is raised.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.17.3 Methods

Methods of the **TIBCStoredProc** class.

For a complete list of the **TIBCStoredProc** class members, see the [TIBCStoredProc Members](#) topic.

### Public

Name	Description
<a href="#">AddWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">AutoCommit</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to automatically commit each update, insert or delete statement by database server.
<a href="#">BaseSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#">BeforeFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#">BeforeUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#">BreakExec</a> (inherited from <a href="#">TCustomDADataset</a> )	Breaks execution of the SQL statement on the server.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.

<a href="#"><u>CancelUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#"><u>CommitUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Clears the cached updates buffer.
<a href="#"><u>Connection</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the connection in which the dataset will be executed.
<a href="#"><u>CreateBlobStream</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#"><u>CreateProcCall</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Assigns PL/SQL block that calls stored procedure to the SQL property.
<a href="#"><u>Cursor</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
<a href="#"><u>Debug</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#"><u>DeferredPost</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Makes permanent changes to the database server.
<a href="#"><u>DeleteWhere</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#"><u>DetailFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#"><u>Disconnected</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to keep dataset opened after connection is closed.
<a href="#"><u>DMLRefresh</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to refresh record by the RETURNING clause when insert is performed.
<a href="#"><u>Encryption</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the options of the data encryption in a dataset.
<a href="#"><u>ExecProc</u></a>	Executes a SQL statement on the server.
<a href="#"><u>Execute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Executes a SQL statement on the server.

<a href="#"><u>Executing</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether SQL statement is still being executed.
<a href="#"><u>Fetches</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to determine if TCustomDADataset has already fetched all rows.
<a href="#"><u>Fetching</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#"><u>FetchingAll</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#"><u>FetchRows</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to define the number of rows to be transferred across the network at the same time.
<a href="#"><u>FilterSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#"><u>FinalSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#"><u>FindKey</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Searches for a record which contains specified field values.
<a href="#"><u>FindMacro</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether a specified macro exists in a dataset.
<a href="#"><u>FindNearest</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#"><u>FindParam</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Determines if a parameter with the specified name exists in a dataset.
<a href="#"><u>GeneratorMode</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify which method is used internally to generate a sequenced field.
<a href="#"><u>GeneratorStep</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.

<a href="#">GetArray</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Retrieves a TIBCArrary object for a field when only its name is known.
<a href="#">GetBlob</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Retrieves a TIBCBlob object for a field when only its name is known.
<a href="#">GetDataType</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#">GetFieldObject</a> (inherited from <a href="#">TCustomDADataset</a> )	Returns a multireference shared object from field.
<a href="#">GetFieldPrecision</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the precision of a number field.
<a href="#">GetFieldScale</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves the scale of a number field.
<a href="#">GetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Retrieves an ORDER BY clause from a SQL statement.
<a href="#">GotoCurrent</a> (inherited from <a href="#">TCustomDADataset</a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">IsQuery</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to check if the SQL statement returns rows.
<a href="#">KeyFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#">KeyGenerator</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.

<a href="#"><u>Locate</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#"><u>LocateEx</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Excludes features that don't need to be included to the <a href="#"><u>TMemDataSet.Locate</u></a> method of TDataSet.
<a href="#"><u>Lock</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Locks the current record.
<a href="#"><u>LockMode</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to indicate when to perform a locking of editing record.
<a href="#"><u>MacroByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Finds a Macro with the name passed in Name.
<a href="#"><u>MacroCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to get the number of macros associated with the Macros property.
<a href="#"><u>Macros</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Makes it possible to change SQL queries easily.
<a href="#"><u>MasterFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#"><u>MasterSource</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the data source component which binds current dataset to the master one.
<a href="#"><u>OnUpdateError</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#"><u>OnUpdateRecord</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when a single update component can not handle the updates.
<a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Called to set or use parameter information for a specific parameter based on its name.
<a href="#"><u>ParamCheck</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.

<a href="#">ParamCount</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate how many parameters are there in the Params property.
<a href="#">Params</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#">Plan</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get or set the PLAN clause of the SELECT statement.
<a href="#">Prepare</a>	Describes the stored procedure parameters.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">PrepareSQL</a>	Describes the stored procedure parameters.
<a href="#">ReadOnly</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#">RefreshOptions</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate when the editing record is refreshed.
<a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )	Actuali es field values for the current record.
<a href="#">RestoreSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">RowsAffected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#">RowsDeleted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were deleted during the last query operation.
<a href="#">RowsFetched</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the number of the currently fetched rows.

<a href="#">RowsInserted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were inserted during the last query operation.
<a href="#">RowsUpdated</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were updated during the last query operation.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataset</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to provide a SQL statement that a query component executes when its Open method is called.
<a href="#">SQLDelete</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.
<a href="#">SQLInsert</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#">SQLLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to perform a record lock.
<a href="#">SQLRefresh</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#">TCustomDADataset.RefreshRecord</a> procedure.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">SQLType</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the typecode of the SQL statement being processed by the InterBase database server.
<a href="#">SQLUpdate</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.



<a href="#">Transaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine the transaction under which the query of this dataset executes.
<a href="#">UniDirectional</a> (inherited from <a href="#">TCustomDADataset</a> )	Used if an application does not need bidirectional access to records in the result set.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.
<a href="#">UpdateTransaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get or set the transaction for modifying a dataset.

### See Also

- [TIBCStoredProc Class](#)
- [TIBCStoredProc Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.17.3.1 ExecProc Method

Executes a SQL statement on the server.

### Class

[TIBCStoredProc](#)

### Syntax

```
procedure ExecProc;
```

### Remarks

ExecProc is similar to the [TCustomDADataset.Execute](#) method. It is included for compatibility with TStoredProc.

## See Also

- [TCustomDADataset.Execute](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.17.3.2 Prepare Method

Describes the stored procedure parameters.

## Class

[TIBCStoredProc](#)

## Syntax

```
procedure Prepare; override;
```

## Remarks

Call the Prepare method to describe the parameters of stored procedure. You can define parameters at design time if ParametersEditor is opened. Prepare method prepares the EXECUTE PROCEDURE statement. To prepare the SELECT statement use the [PrepareSQL](#) method.

## See Also

- [PrepareSQL](#)
  - [TCustomDADataset.Execute](#)
- 

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.17.3.3 PrepareSQL Method

Describes the stored procedure parameters.

## Class

[TIBCStoredProc](#)

## Syntax

```
procedure PrepareSQL(IsQuery: boolean = False);
```

### Parameters

*IsQuery*

True, if the SELECT statement should be prepared.

## Remarks

Use the PrepareSQL method to describe the parameters of stored procedure. The Execute or Open method calls it automatically if it is necessary. You can define the parameters at design time if ParametersEditor is opened. Set IsQuery parameter to True to prepare SELECT statement. Set it to False or omit it to prepare EXECUTE PROCEDURE statement.

## See Also

- [Prepare](#)
- [TCustomDADataset.Execute](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.1 TIBCTable Class

A component for retrieving and updating data in a single table without writing SQL statements.

For a list of all members of this type, see [TIBCTable](#) members.

## Unit

[IBC](#)

## Syntax

```
TIBCTable = class(TCustomIBCTable);
```

## Remarks

The TIBCTable component allows retrieving and updating data in a single table without writing SQL statements. Use TIBCTable to access data in a table . Use the TableName property to specify table name. TIBCTable uses the KeyFields property to build SQL statements for updating table data. KeyFields is a string containing a semicolon-delimited list of the field names.

## Inheritance Hierarchy

```

TMemDataSet
  TCustomDADataset
    TCustomIBCDataset
      TCustomIBCQuery
        TCustomIBCTable
          TIBCTable

```

## See Also

- [Updating Data with IBDAC Dataset Components](#)
- [Master/Detail Relationships](#)
- [TCustomIBCTable](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.18.1 Members

[TIBCTable](#) class overview.

## Properties

Name	Description
------	-------------

<a href="#">AutoCommit</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to automatically commit each update, insert or delete statement by database server.
<a href="#">BaseSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataset</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">Connection</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the connection in which the dataset will be executed.
<a href="#">Cursor</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
<a href="#">Debug</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#">DetailFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#">Disconnected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to keep dataset opened after connection is closed.
<a href="#">DMLRefresh</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to refresh record by the RETURNIG clause when insert is performed.
<a href="#">Encryption</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the options of the data encryption in a dataset.
<a href="#">Exists</a> (inherited from <a href="#">TCustomIBCTable</a> )	Indicates whether a table with the name passed in TableName exists in the database.
<a href="#">FetchAll</a>	Defines whether to request all records of the query from database server when the dataset is being opened.
<a href="#">FetchRows</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to define the number of rows to be transferred across the network at the same time.

<a href="#">FilterSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#">FinalSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#">GeneratorMode</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify which method is used internally to generate a sequenced field.
<a href="#">GeneratorStep</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">IsQuery</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to check if the SQL statement returns rows.
<a href="#">KeyFields</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#">KeyGenerator</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">LockMode</a>	Used to specify what kind of lock will be performed when editing a record.
<a href="#">MacroCount</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to get the number of macros associated with the Macros property.

<a href="#"><u>Macros</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Makes it possible to change SQL queries easily.
<a href="#"><u>MasterFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#"><u>MasterSource</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the data source component which binds current dataset to the master one.
<a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#"><u>ParamCheck</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#"><u>ParamCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate how many parameters are there in the Params property.
<a href="#"><u>Params</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#"><u>Plan</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get or set the PLAN clause of the SELECT statement.
<a href="#"><u>Prepared</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Determines whether a query is prepared for execution or not.
<a href="#"><u>ReadOnly</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#"><u>RefreshOptions</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate when the editing record is refreshed.
<a href="#"><u>RowsAffected</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#"><u>RowsDeleted</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to indicate the number of rows that were deleted during the last query operation.
<a href="#"><u>RowsFetched</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get the number of the currently fetched rows.

<a href="#"><u>RowsInserted</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to indicate the number of rows that were inserted during the last query operation.
<a href="#"><u>RowsUpdated</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to indicate the number of rows that were updated during the last query operation.
<a href="#"><u>SQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to provide a SQL statement that a query component executes when its Open method is called.
<a href="#"><u>SQLDelete</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.
<a href="#"><u>SQLInsert</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#"><u>SQLLock</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify a SQL statement that will be used to perform a record lock.
<a href="#"><u>SQLRefresh</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#"><u>TCustomDADataset.RefreshRecord</u></a> procedure.
<a href="#"><u>SQLType</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get the typecode of the SQL statement being processed by the InterBase database server.
<a href="#"><u>SQLUpdate</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#"><u>TableName</u></a>	Used to specify the name of the database table this component encapsulates.
<a href="#"><u>Transaction</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to determine the transaction under which the query of this dataset executes.
<a href="#"><u>UniDirectional</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used if an application does not need bidirectional access to records in the result set.

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

Used to indicate the update status for the current record when cached updates are enabled.

[UpdatesPending](#) (inherited from [TMemDataSet](#))

Used to check the status of the cached updates buffer.

[UpdateTransaction](#) (inherited from [TCustomIBCDataset](#))

Used to get or set the transaction for modifying a dataset.

## Methods

Name	Description
<a href="#">AddWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">BreakExec</a> (inherited from <a href="#">TCustomDADataset</a> )	Breaks execution of the SQL statement on the server.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">CreateBlobStream</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#">CreateProcCall</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Assigns PL/SQL block that calls stored procedure to the SQL property.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">DeleteTable</a> (inherited from <a href="#">TCustomIBCTable</a> )	Deletes a table from a database.
<a href="#">DeleteWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#">EmptyTable</a> (inherited from <a href="#">TCustomIBCTable</a> )	Truncates the current table content on the server.
<a href="#">Execute</a> (inherited from <a href="#">TCustomDADataset</a> )	Executes a SQL statement on the server.



<a href="#"><u>Executing</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether SQL statement is still being executed.
<a href="#"><u>Fetches</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to determine if TCustomDADataset has already fetched all rows.
<a href="#"><u>Fetching</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is still fetching rows.
<a href="#"><u>FetchingAll</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#"><u>FindKey</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Searches for a record which contains specified field values.
<a href="#"><u>FindMacro</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether a specified macro exists in a dataset.
<a href="#"><u>FindNearest</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#"><u>FindParam</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Determines if a parameter with the specified name exists in a dataset.
<a href="#"><u>GetArray</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Retrieves a TIBCArrary object for a field when only its name is known.
<a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Retrieves a TIBCBlob object for a field when only its name is known.
<a href="#"><u>GetDataType</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#"><u>GetFieldObject</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Returns a multireference shared object from field.
<a href="#"><u>GetFieldPrecision</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves the precision of a number field.
<a href="#"><u>GetFieldScale</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves the scale of a number field.
<a href="#"><u>GetOrderBy</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves an ORDER BY clause from a SQL statement.
<a href="#"><u>GotoCurrent</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Sets the current record in this dataset similar to the current record in another dataset.

<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">Lock</a> (inherited from <a href="#">TCustomDADataset</a> )	Locks the current record.
<a href="#">MacroByName</a> (inherited from <a href="#">TCustomDADataset</a> )	Finds a Macro with the name passed in Name.
<a href="#">ParamByName</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Called to set or use parameter information for a specific parameter based on its name.
<a href="#">Prepare</a> (inherited from <a href="#">TCustomDADataset</a> )	Allocates, opens, and parses cursor for a query.
<a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )	Actuali es field values for the current record.
<a href="#">RestoreSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.

[UnPrepare](#) (inherited from [TMemDataSet](#))

Frees the resources allocated for a previously prepared query on the server and client sides.

[UpdateResult](#) (inherited from [TMemDataSet](#))

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Indicates the current update status for the dataset when cached updates are enabled.

## Events

Name	Description
<a href="#">AfterExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after a component has executed a query to database.
<a href="#">AfterFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after dataset finishes fetching data from server.
<a href="#">AfterUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#">BeforeFetch</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#">BeforeUpdateExecute</a> (inherited from <a href="#">TCustomDADataset</a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.18.2 Properties

Properties of the **TIBCTable** class.

For a complete list of the **TIBCTable** class members, see the [TIBCTable Members](#) topic.

## Public

Name	Description
<a href="#">AddWhere</a> (inherited from <a href="#">TCustomDADataset</a> )	Adds condition to the WHERE clause of SELECT statement in the SQL property.

<a href="#"><u>AfterExecute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs after a component has executed a query to database.
<a href="#"><u>AfterFetch</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs after dataset finishes fetching data from server.
<a href="#"><u>AfterUpdateExecute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs after executing insert, delete, update, lock and refresh operations.
<a href="#"><u>ApplyUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#"><u>AutoCommit</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to automatically commit each update, insert or delete statement by database server.
<a href="#"><u>BaseSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
<a href="#"><u>BeforeFetch</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs before dataset is going to fetch block of records from the server.
<a href="#"><u>BeforeUpdateExecute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Occurs before executing insert, delete, update, lock, and refresh operations.
<a href="#"><u>BreakExec</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Breaks execution of the SQL statement on the server.
<a href="#"><u>CachedUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#"><u>CancelUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#"><u>CommitUpdates</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Clears the cached updates buffer.
<a href="#"><u>Connection</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the connection in which the dataset will be executed.
<a href="#"><u>CreateBlobStream</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
<a href="#"><u>CreateProcCall</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Assigns PL/SQL block that calls stored procedure to the SQL property.

<a href="#"><u>Cursor</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
<a href="#"><u>Debug</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to display executing statement, all its parameters' values, and the type of parameters.
<a href="#"><u>DeferredPost</u></a> (inherited from <a href="#"><u>TMemDataset</u></a> )	Makes permanent changes to the database server.
<a href="#"><u>DeleteTable</u></a> (inherited from <a href="#"><u>TCustomIBCTable</u></a> )	Deletes a table from a database.
<a href="#"><u>DeleteWhere</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
<a href="#"><u>DetailFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
<a href="#"><u>Disconnected</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to keep dataset opened after connection is closed.
<a href="#"><u>DMLRefresh</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to refresh record by the RETURNIG clause when insert is performed.
<a href="#"><u>EmptyTable</u></a> (inherited from <a href="#"><u>TCustomIBCTable</u></a> )	Truncates the current table content on the server.
<a href="#"><u>Encryption</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the options of the data encryption in a dataset.
<a href="#"><u>Execute</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Executes a SQL statement on the server.
<a href="#"><u>Executing</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether SQL statement is still being executed.
<a href="#"><u>Exists</u></a> (inherited from <a href="#"><u>TCustomIBCTable</u></a> )	Indicates whether a table with the name passed in TableName exists in the database.
<a href="#"><u>Fetches</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to determine if TCustomDADataset has already fetched all rows.
<a href="#"><u>Fetching</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is still fetching rows.

<a href="#"><u>FetchingAll</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to learn whether TCustomDADataset is fetching all rows to the end.
<a href="#"><u>FetchRows</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to define the number of rows to be transferred across the network at the same time.
<a href="#"><u>FilterSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to change the WHERE clause of SELECT statement and reopen a query.
<a href="#"><u>FinalSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
<a href="#"><u>FindKey</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Searches for a record which contains specified field values.
<a href="#"><u>FindMacro</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Indicates whether a specified macro exists in a dataset.
<a href="#"><u>FindNearest</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.
<a href="#"><u>FindParam</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Determines if a parameter with the specified name exists in a dataset.
<a href="#"><u>GeneratorMode</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify which method is used internally to generate a sequenced field.
<a href="#"><u>GeneratorStep</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
<a href="#"><u>GetArray</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Retrieves a TIBCArrary object for a field when only its name is known.
<a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Retrieves a TIBCBlob object for a field when only its name is known.
<a href="#"><u>GetDataType</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Returns internal field types defined in the MemData and accompanying modules.
<a href="#"><u>GetFieldObject</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Returns a multireference shared object from field.

<a href="#"><u>GetFieldPrecision</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves the precision of a number field.
<a href="#"><u>GetFieldScale</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves the scale of a number field.
<a href="#"><u>GetOrderBy</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Retrieves an ORDER BY clause from a SQL statement.
<a href="#"><u>GotoCurrent</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Sets the current record in this dataset similar to the current record in another dataset.
<a href="#"><u>Handle</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the handle for the SQL statement of TCustomIBCDataset.
<a href="#"><u>IndexFieldNames</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to check if the SQL statement returns rows.
<a href="#"><u>KeyFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
<a href="#"><u>KeyGenerator</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
<a href="#"><u>LocalConstraints</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#"><u>LocalUpdate</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Used to prevent implicit update of rows on database server.
<a href="#"><u>Locate</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#"><u>LocateEx</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Overloaded. Excludes features that don't need to be included to the <a href="#"><u>TMemDataSet.Locate</u></a> method of TDataSet.
<a href="#"><u>Lock</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Locks the current record.
<a href="#"><u>MacroByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Finds a Macro with the name passed in Name.

<a href="#"><u>MacroCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to get the number of macros associated with the Macros property.
<a href="#"><u>Macros</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Makes it possible to change SQL queries easily.
<a href="#"><u>MasterFields</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
<a href="#"><u>MasterSource</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify the data source component which binds current dataset to the master one.
<a href="#"><u>OnUpdateError</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#"><u>OnUpdateRecord</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Occurs when a single update component can not handle the updates.
<a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to specify the behaviour of the TCustomIBCDataset object.
<a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Called to set or use parameter information for a specific parameter based on its name.
<a href="#"><u>ParamCheck</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
<a href="#"><u>ParamCount</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to indicate how many parameters are there in the Params property.
<a href="#"><u>Params</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Used to view and set parameter names, values, and data types dynamically.
<a href="#"><u>Plan</u></a> (inherited from <a href="#"><u>TCustomIBCDataset</u></a> )	Used to get or set the PLAN clause of the SELECT statement.
<a href="#"><u>Prepare</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )	Allocates, opens, and parses cursor for a query.
<a href="#"><u>Prepared</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )	Determines whether a query is prepared for execution or not.



<a href="#">ReadOnly</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to prevent users from updating, inserting, or deleting data in the dataset.
<a href="#">RefreshOptions</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate when the editing record is refreshed.
<a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )	Actuali es field values for the current record.
<a href="#">RestoreSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Restores the SQL property modified by AddWhere and SetOrderBy.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataset</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">Resync</a> (inherited from <a href="#">TCustomDADataset</a> )	Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataset</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">RowsAffected</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
<a href="#">RowsDeleted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were deleted during the last query operation.
<a href="#">RowsFetched</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the number of the currently fetched rows.
<a href="#">RowsInserted</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were inserted during the last query operation.
<a href="#">RowsUpdated</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to indicate the number of rows that were updated during the last query operation.
<a href="#">SaveSQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Saves the SQL property value to BaseSQL.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataset</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )	Builds an ORDER BY clause of a SELECT statement.

<a href="#">SQL</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to provide a SQL statement that a query component executes when its Open method is called.
<a href="#">SQLDelete</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying a deletion to a record.
<a href="#">SQLInsert</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
<a href="#">SQLLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to perform a record lock.
<a href="#">SQLRefresh</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used to refresh current record by calling the <a href="#">TCustomDADataset.RefreshRecord</a> procedure.
<a href="#">SQLSaved</a> (inherited from <a href="#">TCustomDADataset</a> )	Determines if the <a href="#">SQL</a> property value was saved to the <a href="#">BaseSQL</a> property.
<a href="#">SQLType</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get the typecode of the SQL statement being processed by the InterBase database server.
<a href="#">SQLUpdate</a> (inherited from <a href="#">TCustomDADataset</a> )	Used to specify a SQL statement that will be used when applying an update to a dataset.
<a href="#">Transaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to determine the transaction under which the query of this dataset executes.
<a href="#">UniDirectional</a> (inherited from <a href="#">TCustomDADataset</a> )	Used if an application does not need bidirectional access to records in the result set.
<a href="#">UnLock</a> (inherited from <a href="#">TCustomDADataset</a> )	Releases a record lock.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.

<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.
<a href="#">UpdateTransaction</a> (inherited from <a href="#">TCustomIBCDataset</a> )	Used to get or set the transaction for modifying a dataset.

## Published

Name	Description
<a href="#">FetchAll</a>	Defines whether to request all records of the query from database server when the dataset is being opened.
<a href="#">LockMode</a>	Used to specify what kind of lock will be performed when editing a record.
<a href="#">TableName</a>	Used to specify the name of the database table this component encapsulates.

## See Also

- [TIBCTable Class](#)
- [TIBCTable Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.18.2.1 FetchAll Property

Defines whether to request all records of the query from database server when the dataset is being opened.

## Class

[TIBCTable](#)

## Syntax

```
property FetchAll: boolean;
```

## Remarks

When set to True, all records of the query are requested from database server when the dataset is being opened. When set to False, records are retrieved when a data-aware component or a program requests it. If a query can return a lot of records, set this property to False if initial response time is important.

When the FetchAll property is False, the first call to [TMemDataSet.Locate](#) and [TMemDataSet.LocateEx](#) methods may take a lot of time to retrieve additional

records to the client side.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.18.2.2 LockMode Property

Used to specify what kind of lock will be performed when editing a record.

### Class

[TIBCTable](#)

### Syntax

```
property LockMode: TLockMode default lmNone;
```

### Remarks

Use the LockMode property to define what kind of lock will be performed when editing a record. Locking a record is useful in creating multi-user applications. It prevents modification of a record by several users at the same time. Locking is performed by the RefreshRecord method. The default value is lmNone.

### See Also

- [TIBCStoredProc.LockMode](#)
  - [TIBCQuery.LockMode](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.18.2.3 TableName Property

Used to specify the name of the database table this component encapsulates.

### Class

[TIBCTable](#)

### Syntax

```
property TableName: string;
```

### Remarks

Use the TableName property to specify the name of the database table this component encapsulates. If [TCustomDADataset.Connection](#) is assigned at design time and correct connection settings are provided, select a valid table name from the TableName drop-down list in Object Inspector.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.1 TIBCTransaction Class

A component for managing transactions in an application.  
For a list of all members of this type, see [TIBCTransaction](#) members.

## Unit

[IBC](#)

## Syntax

```
TIBCTransaction = class (TDATransaction) ;
```

## Remarks

The TIBCTransaction component is used to provide discrete transaction control over connection. It can be used for manipulating simple local and global transactions. All components which are dedicated to perform data access, such as TIBCQuery, TIBCSQL, TIBCScript, must have their Transaction property assigned with one of TIBCTransaction instances.

## Inheritance Hierarchy

[TDATransaction](#)

**TIBCTransaction**

## See Also

- [TCustomDAConnection.StartTransaction](#)
- [TCustomDAConnection.Commit](#)
- [TCustomDAConnection.Rollback](#)
- [TIBCCConnection.DefaultTransaction](#)
- [TIBCCConnection.Transactions](#)
- [TCustomIBCDDataSet.Transaction](#)
- [TIBCSQL.Transaction](#)

© 1997-2013 Devart. All Rights Reserved.

17.13.1.19.1 Members

[TIBCTransaction](#) class overview.

## Properties

Name	Description
<a href="#">Active</a>	Determines if the transaction is active or not.
<a href="#">Connections</a>	Used to specify a connection for the given index.
<a href="#">ConnectionsCount</a>	Used to get the number of connections associated with the transaction component.
<a href="#">DefaultCloseAction</a>	Used to specify the transaction behaviour when connection closes.
<a href="#">DefaultConnection</a>	Used to access the default connection of the transaction.

[Handle](#)

Used to specify the handle of the transaction.

[IsolationLevel](#)

Used to get or set the transaction isolation level and access mode.

[Params](#)

Used to access transaction parameters of the transaction parameter buffer.

## Methods

Name	Description
<a href="#">AddConnection</a>	Binds a TCustomDAConnection object with the transaction component.
<a href="#">Commit</a>	Stores all changes of data associated with the transaction to the database server permanently.
<a href="#">CommitRetaining</a>	Stores to the database server all changes of data associated with the transaction permanently and then retains the transaction context.
<a href="#">FindDefaultConnection</a>	Returns the default connection for the transaction.
<a href="#">ReleaseSavepoint</a>	Destroys the specified savepoint without affecting any work that has been performed after its creation.
<a href="#">RemoveConnection</a>	Disassociates the specified connections from the transaction.
<a href="#">Rollback</a>	Rolls back all changes of data associated with the transaction.
<a href="#">RollbackRetaining</a>	Rolls back all data changes associated with the transaction and retains the transaction context.
<a href="#">RollbackSavepoint</a>	Cancels all updates for the current transaction and restores its state up to the moment of the last defined savepoint.

[StartSavepoint](#)

Defines a point in the transaction to which you can roll back later.

[StartTransaction](#) (inherited from [TDATransaction](#)) Begins a new transaction.  
)

**Events**

Name	Description
<a href="#">OnError</a>	Occurs when processing errors that are raised during executing transaction and savepoint control statements such as COMMIT, ROLLBACK, SAVEPOINT, RELEASE SAVEPOINT and others.

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.2 Properties

Properties of the **TIBCTransaction** class.  
For a complete list of the **TIBCTransaction** class members, see the [TIBCTransaction Members](#) topic.

**Public**

Name	Description
<a href="#">Commit</a> (inherited from <a href="#">TDATransaction</a> )	Commits the current transaction.
<a href="#">Connections</a>	Used to specify a connection for the given index.
<a href="#">ConnectionsCount</a>	Used to get the number of connections associated with the transaction component.
<a href="#">Handle</a>	Used to specify the handle of the transaction.
<a href="#">OnError</a> (inherited from <a href="#">TDATransaction</a> )	Used to process errors that occur during executing a transaction.
<a href="#">Rollback</a> (inherited from <a href="#">TDATransaction</a> )	Discards all modifications of data associated with the current transaction and ends the transaction.
<a href="#">StartTransaction</a> (inherited from <a href="#">TDATransaction</a> )	Begins a new transaction. )

**Published**

Name	Description
------	-------------

[Active](#)

Determines if the transaction is active or not.

[DefaultCloseAction](#)

Used to specify the transaction behaviour when connection closes.

[DefaultConnection](#)

Used to access the default connection of the transaction.

[IsolationLevel](#)

Used to get or set the transaction isolation level and access mode.

[Params](#)

Used to access transaction parameters of the transaction parameter buffer.

### See Also

- [TIBCTransaction Class](#)
  - [TIBCTransaction Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.19.2.1 Active Property

Determines if the transaction is active or not.

### Class

[TIBCTransaction](#)

### Syntax

```
property Active: Boolean stored IsActiveStored default False;
```

### Remarks

The Active property is used to indicate whether transaction is active or not.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.19.2.2 Connections Property(Indexer)

Used to specify a connection for the given index.

### Class

[TIBCTransaction](#)

### Syntax

```
property Connections[Index: integer]: TIBConnection;
```

### Parameters

*Index*

Holds the index for which to specify a connection.



## Remarks

Use the Connections property to specify a connection for the given index.

## See Also

- [ConnectionsCount](#)
- [AddConnection](#)
- [RemoveConnection](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.19.2.3 ConnectionsCount Property

Used to get the number of connections associated with the transaction component.

## Class

[TIBCTransaction](#)

## Syntax

```
property ConnectionsCount: integer;
```

## Remarks

Use the ConnectionsCount property for getting the number of connections associated with the transaction component.

## See Also

- [AddConnection](#)
- [RemoveConnection](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.19.2.4 DefaultCloseAction Property

Used to specify the transaction behaviour when connection closes.

## Class

[TIBCTransaction](#)

## Syntax

```
property DefaultCloseAction: TIBCTransactionAction default  
taRollback;
```

## Remarks

Use the DefaultCloseAction property to specify the transaction behaviour when connection closes.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.2.5 DefaultConnection Property

Used to access the default connection of the transaction.

**Class**

[TIBCTransaction](#)

**Syntax**

```
property DefaultConnection: TIBConnection stored  
IsInternalTrStored;
```

**Remarks**

Use the DefaultConnection property to access the default connection of the transaction.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.2.6 Handle Property

Used to specify the handle of the transaction.

**Class**

[TIBCTransaction](#)

**Syntax**

```
property Handle: TISC_TR_HANDLE;
```

**Remarks**

Use the Handle property to specify the handle of the transaction. Use Handle Property to make calls directly to the InterBase API. Some of the InterBase API functions require a transaction handle as an argument.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.2.7 IsolationLevel Property

Used to get or set the transaction isolation level and access mode.

**Class**

[TIBCTransaction](#)

**Syntax**

```
property IsolationLevel: TIBCIsoationLevel default  
iblReadCommitted;
```

**Remarks**

Use the IsolationLevel property to get or set the transaction isolation level and access mode. You should add IBCClasses unit to the uses list to use this property.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.2.8 Params Property

Used to access transaction parameters of the transaction parameter buffer.

**Class**

[TIBCTransaction](#)

**Syntax**

```
property Params: TStrings;
```

**Remarks**

Use the Params property to access transaction parameters of the transaction parameter buffer. Refer to InterBase API Guide for more information on this parameters.

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.3 Methods

Methods of the **TIBCTransaction** class.

For a complete list of the **TIBCTransaction** class members, see the [TIBCTransaction Members](#) topic.

**Public**

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TDATransaction</a> )	Used to determine if the transaction is active.
<a href="#">AddConnection</a>	Binds a TCustomDAConnection object with the transaction component.
<a href="#">Commit</a>	Stores all changes of data associated with the transaction to the database server permanently.
<a href="#">CommitRetaining</a>	Stores to the database server all changes of data associated with the transaction permanently and then retains the transaction context.
<a href="#">DefaultCloseAction</a> (inherited from <a href="#">TDATransaction</a> )	Used to specify the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.
<a href="#">FindDefaultConnection</a>	Returns the default connection for the transaction.

[OnError](#) (inherited from [TDATransaction](#))

Used to process errors that occur during executing a transaction.

[ReleaseSavepoint](#)

Destroys the specified savepoint without affecting any work that has been performed after its creation.

[RemoveConnection](#)

Disassociates the specified connections from the transaction.

[Rollback](#)

Rolls back all changes of data associated with the transaction.

[RollbackRetaining](#)

Rolls back all data changes associated with the transaction and retains the transaction context.

[RollbackSavepoint](#)

Cancels all updates for the current transaction and restores its state up to the moment of the last defined savepoint.

[StartSavepoint](#)

Defines a point in the transaction to which you can roll back later.

[StartTransaction](#) (inherited from [TDATransaction](#))  
) Begins a new transaction.

### See Also

- [TIBCTransaction Class](#)
- [TIBCTransaction Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.1.19.3.1 AddConnection Method

Binds a TCustomDAConnection object with the transaction component.

### Class

[TIBCTransaction](#)

### Syntax

```
function AddConnection(Connection: TIBCCConnection): integer;
```

### Parameters

*Connection*

Holds a TCustomDAConnection object to associate with the transaction component.

### Return Value

the index of associated connection in the connection list.

**Remarks**

Use the AddConnection method to associate a TCustomDAConnection object with the transaction component.

**See Also**

- [RemoveConnection](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.3.2 Commit Method

Stores all changes of data associated with the transaction to the database server permanently.

**Class**

[TIBCTransaction](#)

**Syntax**

```
procedure Commit; override;
```

**Remarks**

Call the Commit method to store to the database server all changes of data associated with the transaction permanently.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.3.3 CommitRetaining Method

Stores to the database server all changes of data associated with the transaction permanently and then retains the transaction context.

**Class**

[TIBCTransaction](#)

**Syntax**

```
procedure CommitRetaining;
```

**Remarks**

Call the CommitRetaining method to store to the database server all changes of data associated with the transaction permanently and then retain the transaction context.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.3.4 FindDefaultConnection Method

Returns the default connection for the transaction.

**Class**

[TIBCTransaction](#)**Syntax**

```
function FindDefaultConnection: TIBConnection;
```

**Remarks**

Call the FindDefaultConnection method to return the default connection for the transaction.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.3.5 ReleaseSavepoint Method

Destroys the specified savepoint without affecting any work that has been performed after its creation.

**Class**[TIBCTransaction](#)**Syntax**

```
procedure ReleaseSavepoint(const Name: string);
```

**Parameters**

*Name*

Holds the savepoint name.

**Remarks**

Call the ReleaseSavepoint method to destroy the specified savepoint without affecting any work that has been performed after its creation.

**See Also**

- [StartSavepoint](#)
  - [RollbackSavepoint](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.3.6 RemoveConnection Method

Disassociates the specified connections from the transaction.

**Class**[TIBCTransaction](#)**Syntax**

```
procedure RemoveConnection(Connection: TIBConnection);
```

**Parameters**

*Connection*

Holds the connections to disassociate.

**Remarks**

Use the RemoveConnection method to disassociate the specified connections from the transaction.

**See Also**

- [AddConnection](#)

---

© 1997-2013 Devart. All Rights Reserved.

**17.13.1.19.3.7 Rollback Method**

Rolls back all changes of data associated with the transaction.

**Class**

[TIBCTransaction](#)

**Syntax**

```
procedure Rollback; override;
```

**Remarks**

Call the Rollback method to roll back all changes of data associated with the transaction.

---

© 1997-2013 Devart. All Rights Reserved.

**17.13.1.19.3.8 RollbackRetaining Method**

Rolls back all data changes associated with the transaction and retains the transaction context.

**Class**

[TIBCTransaction](#)

**Syntax**

```
procedure RollbackRetaining;
```

**Remarks**

Call the RollbackRetaining method to roll back all changes of data associated with the transaction and retain the transaction context.

---

© 1997-2013 Devart. All Rights Reserved.

**17.13.1.19.3.9 RollbackSavepoint Method**

Cancels all updates for the current transaction and restores its state up to the moment of the last defined savepoint.

**Class**

[TIBCTransaction](#)

**Syntax**

```
procedure RollbackSavepoint(const Name: string);
```

**Parameters**

*Name*

Holds the defined savepoint name.

**Remarks**

Call the RollbackSavepoint to cancel all updates for the current transaction and restore its state up to the moment of the last defined savepoint.

**See Also**

- [Rollback](#)
  - [StartSavepoint](#)
  - [ReleaseSavepoint](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.3.10 StartSavepoint Method

Defines a point in the transaction to which you can roll back later.

**Class**

[TIBCTransaction](#)

**Syntax**

```
procedure StartSavepoint(const Name: string);
```

**Parameters**

*Name*

Holds the savepoint name.

**Remarks**

Call the StartSavepoint method to define a point in the transaction to which you can roll back later. As the parameter, you can pass any valid name to identify the savepoint.

To roll back to the last savepoint call [RollbackSavepoint](#).

**See Also**

- [RollbackSavepoint](#)
  - [ReleaseSavepoint](#)
- 

© 1997-2013 Devart. All Rights Reserved.



## 17.13.1.19.4 Events

Events of the **TIBCTransaction** class.

For a complete list of the **TIBCTransaction** class members, see the [TIBCTransaction Members](#) topic.

**Public**

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TDATransaction</a> )	Used to determine if the transaction is active.
<a href="#">Commit</a> (inherited from <a href="#">TDATransaction</a> )	Commits the current transaction.
<a href="#">DefaultCloseAction</a> (inherited from <a href="#">TDATransaction</a> )	Used to specify the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.
<a href="#">Rollback</a> (inherited from <a href="#">TDATransaction</a> )	Discards all modifications of data associated with the current transaction and ends the transaction.
<a href="#">StartTransaction</a> (inherited from <a href="#">TDATransaction</a> )	Begins a new transaction.

**Published**

Name	Description
<a href="#">OnError</a>	Occurs when processing errors that are raised during executing transaction and savepoint control statements such as COMMIT, ROLLBACK, SAVEPOINT, RELEASE SAVEPOINT and others.

**See Also**

- [TIBCTransaction Class](#)
- [TIBCTransaction Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.13.1.19.4.1 OnError Event

Occurs when processing errors that are raised during executing transaction and savepoint control statements such as COMMIT, ROLLBACK, SAVEPOINT, RELEASE SAVEPOINT and others.

**Class**

[TIBCTransaction](#)

## Syntax

```
property OnError: TIBCTransactionErrorEvent;
```

## Remarks

Write the OnError event handler to process errors that occur during executing transaction and savepoint control statements such as COMMIT, ROLLBACK, SAVEPOINT, RELEASE SAVEPOINT and others. Check the E parameter to get an error code.

**Note:** You should explicitly add [IBCErrors](#) unit to 'uses' list to use OnError event handler.

This event occur only when two or more connections are associated with the transaction. When only one connection is assigned to the transaction, then the OnError event of the TIBConnection class arises.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.2 TIBCUUpdateSQL Class

A component for tuning update operations for the DataSet component. For a list of all members of this type, see [TIBCUUpdateSQL](#) members.

## Unit

[IBC](#)

## Syntax

```
TIBCUUpdateSQL = class (TCustomDAUpdateSQL);
```

## Remarks

Use the TIBCUUpdateSQL component to provide DML statements for the dataset components that return read-only result set. This component also allows setting objects that can be used for executing update operations. You may prefer to use directly SQLInsert, SQLUpdate, and SQLDelete properties of the [TCustomDADataSet](#) descendants.

## Inheritance Hierarchy

[TCustomDAUpdateSQL](#)  
**TIBCUUpdateSQL**

© 1997-2013 Devart. All Rights Reserved.

### 17.13.1.20.1 Members

[TIBCUUpdateSQL](#) class overview.

## Properties

Name	Description
<a href="#">DataSet</a> (inherited from <a href="#">TCustomDAUpdateSQL</a> )	Used to hold a reference to the TCustomDADataSet object that is being updated.

[DeleteObject](#) (inherited from [TCustomDAUpdateSQL](#))

[DeleteSQL](#) (inherited from [TCustomDAUpdateSQL](#))

[InsertObject](#) (inherited from [TCustomDAUpdateSQL](#))

[InsertSQL](#) (inherited from [TCustomDAUpdateSQL](#))

[LockObject](#) (inherited from [TCustomDAUpdateSQL](#))

[LockSQL](#) (inherited from [TCustomDAUpdateSQL](#))

[ModifyObject](#) (inherited from [TCustomDAUpdateSQL](#))

[ModifySQL](#) (inherited from [TCustomDAUpdateSQL](#))

[RefreshObject](#) (inherited from [TCustomDAUpdateSQL](#))

[RefreshSQL](#) (inherited from [TCustomDAUpdateSQL](#))

[SQL](#) (inherited from [TCustomDAUpdateSQL](#))

Provides ability to perform advanced adjustment of the delete operations.

Used when deleting a record.

Provides ability to perform advanced adjustment of insert operations.

Used when inserting a record.

Provides ability to perform advanced adjustment of lock operations.

Used to lock the current record.

Provides ability to perform advanced adjustment of modify operations.

Used when updating a record.

Provides ability to perform advanced adjustment of refresh operations.

Used to specify an SQL statement that will be used for refreshing the current record by

[TCustomDADataset.RefreshRecord](#) procedure.

Used to return a SQL statement for one of the ModifySQL, InsertSQL, or DeleteSQL properties.

## Methods

Name	Description
<a href="#">Apply</a> (inherited from <a href="#">TCustomDAUpdateSQL</a> )	Sets parameters for a SQL statement and executes it to update a record.
<a href="#">ExecSQL</a> (inherited from <a href="#">TCustomDAUpdateSQL</a> )	Executes a SQL statement.

© 1997-2013 Devart. All Rights Reserved.

## 17.13.2 Types

Types in the **IBC** unit.

## Types

Name	Description
<a href="#">TIBCTransactionErrorEvent</a>	This type is used for the <a href="#">TIBCTransaction.OnError</a> event.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.2.1 TIBCTransactionErrorEvent Procedure Reference

This type is used for the [TIBCTransaction.OnError](#) event.

##### Unit

[IBC](#)

##### Syntax

```
TIBCTransactionErrorEvent = procedure (Sender: TObject; E: EIBCErrors) of object;
```

##### Parameters

*Sender*

An object that raised the event.

*E*

Holds the error code.

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.3 Enumerations

Enumerations in the **IBC** unit.

##### Enumerations

Name	Description
<a href="#">TGeneratorMode</a>	Specifies the method used internally to generate a sequenced field.
<a href="#">TIBCTransactionProtocol</a>	Specifies the network protocol of connection with InterBase server.
<a href="#">TIBCTransactionAction</a>	Specifies the transaction behaviour when connection closes.

© 1997-2013 Devart. All Rights Reserved.

##### 17.13.3.1 TGeneratorMode Enumeration

Specifies the method used internally to generate a sequenced field.

##### Unit

[IBC](#)

## Syntax

```
TGeneratorMode = (gmInsert, gmPost);
```

## Values

Value	Meaning
<b>gmInsert</b>	New record is inserted into the dataset where the first key field populated with a sequenced value. Application may modify this field before posting the record to the database.
<b>gmPost</b>	Database server populates the key field with a sequenced value when application posts the record to the database. However if user specified the key field value, the key generator will not be used.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.3.2 TIBCPProtocol Enumeration

Specifies the network protocol of connection with InterBase server.

## Unit

[IBC](#)

## Syntax

```
TIBCPProtocol = (TCP, NetBEUI, SPX);
```

## Values

Value	Meaning
<b>NetBEUI</b>	Uses the NetBEUI protocol.
<b>SPX</b>	Uses the SPX protocol.
<b>TCP</b>	Uses the TCP protocol. The default value.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.3.3 TIBCTransactionAction Enumeration

Specifies the transaction behaviour when connection closes.

## Unit

[IBC](#)

## Syntax

```
TIBCTransactionAction = (taCommit, taRollback, taCommitRetaining, taRollbackRetaining);
```

## Values

Value	Meaning
<b>taCommit</b>	Transaction is committed and is closed.

**taCommitRetain** Transaction is committed and remains opened.  
**ng**  
**taRollback** Transaction is rolled back and is closed.  
**taRollbackRetain** Transaction is rolled back and remains opened.  
**ng**

© 1997-2013 Devart. All Rights Reserved.

## 17.13.4 Routines

Routines in the **IBC** unit.

### Routines

Name	Description
<a href="#">DefaultConnection</a>	Call this function to get pointer to default connection object.

© 1997-2013 Devart. All Rights Reserved.

### 17.13.4.1 DefaultConnection Function

Call this function to get pointer to default connection object.

#### Unit

[IBC](#)

#### Syntax

```
function DefaultConnection: TIBCCConnection;
```

© 1997-2013 Devart. All Rights Reserved.

## 17.13.5 Variables

Variables in the **IBC** unit.

### Variables

Name	Description
<a href="#">Connections</a>	Holds pointers to all TIBCCConnection objects of an application.
<a href="#">DefConnection</a>	Read this variable to get pointer to default connection object. Same as DefaultConnection function.
<a href="#">UseDefConnection</a>	When set to true enables TCustomIBCDataset and TIBCSQL components to use default connection if they are not attached to any connection.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.5.1 Connections Variable

Holds pointers to all TIBCCConnection objects of an application.

##### Unit

[IBC](#)

##### Syntax

```
Connections: TConnectionList;
```

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.5.2 DefConnection Variable

Read this variable to get pointer to default connection object. Same as DefaultConnection function.

##### Unit

[IBC](#)

##### Syntax

```
DefConnection: TIBCCConnection;
```

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.5.3 UseDefConnection Variable

When set to true enables TCustomIBCDataset and TIBCSQL components to use default connection if they are not attached to any connection.

##### Unit

[IBC](#)

##### Syntax

```
UseDefConnection: boolean;
```

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.13.6 Constants

Constants in the **IBC** unit.

##### Constants

Name	Description
<a href="#">IBDACVersion</a>	Read this constant to get current version number for IBDAC.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.13.6.1 IBDACVersion Constant

Read this constant to get current version number for IBDAC.

#### Unit

[IBC](#)

#### Syntax

```
IBDACVersion = '4.6.12';
```

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14 IBCAdmin

This unit contains implementation of components, used for InterBase/Firebird server administration.

### Classes

Name	Description
<a href="#">TCustomIBCSERVICE</a>	TCustomIBCSERVICE is the ancestor object from which all TIBCSERVICE components descend.
<a href="#">TIBCBACKUPRESTORESERVICE</a>	TIBCBACKUPRESTORESERVICE is the base class from which <a href="#">TIBCBACKUPSERVICE</a> and <a href="#">TIBCRESTORESERVICE</a> are derived.
<a href="#">TIBCBACKUPSERVICE</a>	Used to backup a database.
<a href="#">TIBCCONFIGPARAMS</a>	TIBCCONFIGPARAMS holds the configuration information about an InterBase server.
<a href="#">TIBCCONFIGSERVICE</a>	Use a TIBCONFIGSERVICE object to configure database parameters.
<a href="#">TIBCCONTROLANDQUERYSERVICE</a>	TIBCCONTROLANDQUERYSERVICE is the base class from which the log, statistical, validation, security, and backup and restore TIBCSERVICE components descend.
<a href="#">TIBCDATABASEINFO</a>	Describes an InterBase database.
<a href="#">TIBJOURNALINFORMATION</a>	A class used to access the TIBCCONFIGSERVICE component properties.
<a href="#">TIBCLICENSEINFO</a>	Stores information about licensed users.



<a href="#">TIBCLicenseMaskInfo</a>	Indicates the software activation certificate options enabled on the server.
<a href="#">TIBCLicensingService</a>	TIBCLicensingService configures the licensing parameters
<a href="#">TIBCLimboTransactionInfo</a>	TIBCLimboTransactionInfo stores information about a limbo transaction.
<a href="#">TIBCLogService</a>	Returns the contents of the interbase.log file from server.
<a href="#">TIBCRestoreService</a>	Used to restore a database.
<a href="#">TIBCSecurityService</a>	Used to manage user access to the InterBase server.
<a href="#">TIBCServerProperties</a>	A class that returns database server information.
<a href="#">TIBCStatisticalService</a>	TIBCStatisticalService is used to view database statistics.
<a href="#">TIBCTraceService</a>	This component is used for working with trace service added in Firebird 2.5.
<a href="#">TIBCUUserInfo</a>	TIBCUUserInfo stores information about an InterBase user for the security service.
<a href="#">TIBCVValidationService</a>	Used to validate a database and reconcile database transactions.
<a href="#">TIBCVersionInfo</a>	Represents the version information about an InterBase server.

## Types

Name	Description
<a href="#">TIBCBackupOptions</a>	Represents the set of <a href="#">TIBCBackupOption</a> .
<a href="#">TIBCNBackupOptions</a>	Represents the set of <a href="#">TIBCNBackupOption</a> .
<a href="#">TIBCRestoreOptions</a>	Represents the set of <a href="#">TIBCRestoreOption</a> .
<a href="#">TIBCStatOptions</a>	Represents the set of <a href="#">TIBCStatOption</a> .
<a href="#">TIBCValidateOptions</a>	Represents the set of <a href="#">TIBCValidateOption</a> .

## Enumerations

Name	Description
<a href="#">TIBCBackupOption</a>	Allows you to build backup options into your application.
<a href="#">TIBCLicensingAction</a>	Allows to add or remove an InterBase software activation certificate.
<a href="#">TIBCNBackupOption</a>	Options used in <a href="#">TIBCBackupRestoreService</a> for nBackup.
<a href="#">TIBCRestoreOption</a>	Specifies the data restore parameters.
<a href="#">TIBCSecurityAction</a>	Specify the type of the operation for InterBase Security Service to perform.
<a href="#">TIBCStatOption</a>	Allows to specify the way the database statistics would be requested.
<a href="#">TIBCTransactionAdvise</a>	Allows to specify the suggested action for ending the transaction.
<a href="#">TIBCTransactionGlobalAction</a>	Determines which action to take concerning limbo transactions.
<a href="#">TIBCTransactionState</a>	Allows to specify the current state of the limbo transaction.
<a href="#">TIBCValidateOption</a>	Sets the options of validation.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1 Classes

Classes in the **IBCAAdmin** unit.

#### Classes

Name	Description
<a href="#">TCustomIBCSERVICE</a>	TCustomIBCSERVICE is the ancestor object from which all TIBCSERVICE components descend.
<a href="#">TIBCBackupRestoreService</a>	TIBCBackupRestoreService is the base class from which <a href="#">TIBCBackupService</a> and <a href="#">TIBCRestoreService</a> are derived.
<a href="#">TIBCBackupService</a>	Used to backup a database.

<a href="#"><u>TIBConfigParams</u></a>	TIBConfigParams holds the configuration information about an InterBase server.
<a href="#"><u>TIBConfigService</u></a>	Use a TIBConfigService object to configure database parameters.
<a href="#"><u>TIBControlAndQueryService</u></a>	TIBControlAndQueryService is the base class from which the log, statistical, validation, security, and backup and restore TIBService components descend.
<a href="#"><u>TIBDatabaseInfo</u></a>	Describes an InterBase database.
<a href="#"><u>TIBJournalInformation</u></a>	A class used to access the TIBConfigService component properties.
<a href="#"><u>TIBLicenseInfo</u></a>	Stores information about licensed users.
<a href="#"><u>TIBLicenseMaskInfo</u></a>	Indicates the software activation certificate options enabled on the server.
<a href="#"><u>TIBLicensingService</u></a>	TIBLicensingService configures the licensing parameters
<a href="#"><u>TIBLimboTransactionInfo</u></a>	TIBLimboTransactionInfo stores information about a limbo transaction.
<a href="#"><u>TIBLogService</u></a>	Returns the contents of the interbase.log file from server.
<a href="#"><u>TIBRestoreService</u></a>	Used to restore a database.
<a href="#"><u>TIBSecurityService</u></a>	Used to manage user access to the InterBase server.
<a href="#"><u>TIBServerProperties</u></a>	A class that returns database server information.
<a href="#"><u>TIBStatisticalService</u></a>	TIBStatisticalService is used to view database statistics.
<a href="#"><u>TIBTraceService</u></a>	This component is used for working with trace service added in Firebird 2.5.
<a href="#"><u>TIBUserInfo</u></a>	TIBUserInfo stores information about an InterBase user for the security service.

[TIBCValidationService](#)

Used to validate a database and reconcile database transactions.

[TIBCVersionInfo](#)

Represents the version information about an InterBase server.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1 TCustomIBCSERVICE Class

TCustomIBCSERVICE is the ancestor object from which all TIBCSERVICE components descend.

For a list of all members of this type, see [TCustomIBCSERVICE](#) members.

#### Unit

[IBCAAdmin](#)

#### Syntax

```
TCustomIBCSERVICE = class (TComponent);
```

#### Remarks

TCustomIBCSERVICE is the ancestor object from which all TIBCSERVICE components descend.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1.1 Members

[TCustomIBCSERVICE](#) class overview.

#### Properties

Name	Description
<a href="#">Active</a>	Used to set the service to active or inactive (default).
<a href="#">Handle</a>	Used to return the database handle.
<a href="#">LoginPrompt</a>	Used to display a login prompt before attaching to a database.
<a href="#">Params</a>	Used to set or return database parameters.
<a href="#">Protocol</a>	Used to select the network protocol.
<a href="#">Server</a>	Used to set the name of the server on which the services are to be run.
<a href="#">ServiceParamBySPB</a>	Used to return and sets SPB parameters.

## Methods

Name	Description
<a href="#">Attach</a>	Attaches to the database.
<a href="#">Detach</a>	Detaches from the database.
<a href="#">ServiceStart</a>	Starts the service.

## Events

Name	Description
<a href="#">OnAttach</a>	Occurs when the database is attached.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.1.2 Properties

Properties of the **TCustomIBCSERVICE** class.

For a complete list of the **TCustomIBCSERVICE** class members, see the [TCustomIBCSERVICE Members](#) topic.

## Public

Name	Description
<a href="#">Handle</a>	Used to return the database handle.
<a href="#">ServiceParamBySPB</a>	Used to return and sets SPB parameters.

## Published

Name	Description
<a href="#">Active</a>	Used to set the service to active or inactive (default).
<a href="#">LoginPrompt</a>	Used to display a login prompt before attaching to a database.
<a href="#">Params</a>	Used to set or return database parameters.
<a href="#">Protocol</a>	Used to select the network protocol.
<a href="#">Server</a>	Used to set the name of the server on which the services are to be run.

## See Also

- [TCustomIBCSERVICE Class](#)
- [TCustomIBCSERVICE Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.1.2.1 Active Property

Used to set the service to active or inactive (default).

**Class**

[TCustomIBCSERVICE](#)

**Syntax**

```
property Active: Boolean default False;
```

**Remarks**

Use the Active property to set the service to active (True) or the default, inactive (False).

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.1.2.2 Handle Property

Used to return the database handle.

**Class**

[TCustomIBCSERVICE](#)

**Syntax**

```
property Handle: TISC_SVC_HANDLE;
```

**Remarks**

Use the Handle property to return the database handle.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.1.2.3 LoginPrompt Property

Used to display a login prompt before attaching to a database.

**Class**

[TCustomIBCSERVICE](#)

**Syntax**

```
property LoginPrompt: Boolean default True;
```

**Remarks**

Use the LoginPrompt property to display (or not display) a login prompt before attaching to a database.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.1.2.4 Params Property

Used to set or return database parameters.

**Class**

[TCustomIBCSERVICE](#)

### Syntax

```
property Params: TStrings;
```

### Remarks

Use the Params property to set or return database parameters.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1.2.5 Protocol Property

Used to select the network protocol.

### Class

[TCustomIBCSERVICE](#)

### Syntax

```
property Protocol: TIBCPROTOCOL default TCP;
```

### Remarks

Use the Protocol property to select the network protocol.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1.2.6 Server Property

Used to set the name of the server on which the services are to be run.

### Class

[TCustomIBCSERVICE](#)

### Syntax

```
property Server: string;
```

### Remarks

Use the Server property to set the name of the server on which the services are to be run.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1.2.7 ServiceParamBySPB Property(Indexer)

Used to return and sets SPB parameters.

### Class

[TCustomIBCSERVICE](#)

### Syntax

```
property ServiceParamBySPB[const Idx: Integer]: string;  
Parameters
```

*Idx*

Holds the index of parameter.

### Remarks

Use the ServiceParamBySPB property to inspect and set SPB parameters.  
For example, DBParamBySPB[isc SPB user name] sets and inspects the user name.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1.3 Methods

Methods of the **TCustomIBCSERVICE** class.

For a complete list of the **TCustomIBCSERVICE** class members, see the [TCustomIBCSERVICE Members](#) topic.

### Public

Name	Description
<a href="#">Attach</a>	Attaches to the database.
<a href="#">Detach</a>	Detaches from the database.
<a href="#">ServiceStart</a>	Starts the service.

### See Also

- [TCustomIBCSERVICE Class](#)
  - [TCustomIBCSERVICE Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1.3.1 Attach Method

Attaches to the database.

### Class

[TCustomIBCSERVICE](#)

### Syntax

```
procedure Attach;
```

### Remarks

Call the Attach method to attach to the database.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1.3.2 Detach Method

Detaches from the database.

### Class

[TCustomIBCSERVICE](#)

### Syntax



```
procedure Detach;
```

### Remarks

Call the Detach method to detach from the database.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1.3.3 ServiceStart Method

Starts the service.

### Class

[TCustomIBCSERVICE](#)

### Syntax

```
procedure ServiceStart; virtual;
```

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1.4 Events

Events of the **TCustomIBCSERVICE** class.

For a complete list of the **TCustomIBCSERVICE** class members, see the [TCustomIBCSERVICE Members](#) topic.

### Published

Name	Description
<a href="#">OnAttach</a>	Occurs when the database is attached.

### See Also

- [TCustomIBCSERVICE Class](#)
- [TCustomIBCSERVICE Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1.4.1 OnAttach Event

Occurs when the database is attached.

### Class

[TCustomIBCSERVICE](#)

### Syntax

```
property OnAttach: TNotifyEvent;
```

### Remarks

Write an OnAttach event handler to take specific actions when a database is attached. If an exception is raised in this event, the database is not attached.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.2 TIBCBBackupRestoreService Class

TIBCBBackupRestoreService is the base class from which [TIBCBBackupService](#) and [TIBCBRestoreService](#) are derived.

For a list of all members of this type, see [TIBCBBackupRestoreService](#) members.

#### Unit

[IBCAAdmin](#)

#### Syntax

```
TIBCBBackupRestoreService = class (TIBCCControlAndQueryService) ;
```

#### Remarks

TIBCBBackupRestoreService is the base class from which [TIBCBBackupService](#) and [TIBCBRestoreService](#) are derived.

**Note:** You should install InterBase 6 to use this feature.

#### Inheritance Hierarchy

[TCustomIBCSERVICE](#)

[TIBCCControlAndQueryService](#)

**TIBCBBackupRestoreService**

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.2.1 Members

[TIBCBBackupRestoreService](#) class overview.

#### Properties

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BackupFile</a>	Holds the path of the backup file name.
<a href="#">BufferSize</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">Eof</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">NBackupLevel</a>	Used to set backup level for nBackup.
<a href="#">NBackupOptions</a>	Used to set backup options for nBackup.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.

<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">UseNBackup</a>	Used to enable or disable using nBackup service.
<a href="#">Verbose</a>	Used to set or return the backup or restore in the verbose mode.

## Methods

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next line of data.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

## Events

Name	Description
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.2.2 Properties

Properties of the **TIBCBBackupRestoreService** class.  
For a complete list of the **TIBCBBackupRestoreService** class members, see the [TIBCBBackupRestoreService Members](#) topic.

## Public

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">BackupFile</a>	Holds the path of the backup file name.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Eof</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.

<a href="#">GetNextLine</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next line of data.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">NBackupLevel</a>	Used to set backup level for nBackup.
<a href="#">NBackupOptions</a>	Used to set backup options for nBackup.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.
<a href="#">UseNBackup</a>	Used to enable or disable using nBackup service.
<a href="#">Verbose</a>	Used to set or return the backup or restore in the verbose mode.

## Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.

## See Also

- [TIBCBBackupRestoreService Class](#)
- [TIBCBBackupRestoreService Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.2.2.1 BackupFile Property

Holds the path of the backup file name.

## Class

[TIBCBBackupRestoreService](#)

**Syntax**

```
property BackupFile: TStrings;
```

**Remarks**

The BackupFile property holds the path of the backup file name.

---

© 1997-2013 Devart. All Rights Reserved.

**17.14.1.2.2.2 NBackupLevel Property**

Used to set backup level for nBackup.

**Class**

[TIBCBBackupRestoreService](#)

**Syntax**

```
property NBackupLevel: integer default 0;
```

**Remarks**

Use the NBackupLevel property to set backup level for nBackup. This property is used only when UseNBackup = True.

---

© 1997-2013 Devart. All Rights Reserved.

**17.14.1.2.2.3 NBackupOptions Property**

Used to set backup options for nBackup.

**Class**

[TIBCBBackupRestoreService](#)

**Syntax**

```
property NBackupOptions: TIBCNBackupOptions default [];
```

**Remarks**

Use the NBackupOptions property to set backup options for nBackup. This property is used only when UseNBackup = True.

---

© 1997-2013 Devart. All Rights Reserved.

**17.14.1.2.2.4 UseNBackup Property**

Used to enable or disable using nBackup service.

**Class**

[TIBCBBackupRestoreService](#)

**Syntax**

```
property UseNBackup: boolean default False;
```

**Remarks**

Use the UseNBackup property to enable or disable using nBackup service. Set this property to True to use nBackup service instead of the standard backup/restore service.

**Note:** nBackup service is supported starting with Firebird 2.5.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.2.2.5 Verbose Property

Used to set or return the backup or restore in the verbose mode.

### Class

[TIBCBackupRestoreService](#)

### Syntax

```
property Verbose: Boolean default False;
```

### Remarks

Use the Verbose property to set or return the backup or restore in the verbose mode.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.3 TIBCBackupService Class

Used to backup a database.

For a list of all members of this type, see [TIBCBackupService](#) members.

### Unit

[IBCAdmin](#)

### Syntax

```
TIBCBackupService = class(TIBCBackupRestoreService);
```

### Remarks

Use a TIBCBackupService object to backup a database.

### Inheritance Hierarchy

[TCustomIBCSERVICE](#)  
[TIBControlAndQueryService](#)  
[TIBCBackupRestoreService](#)  
**TIBCBackupService**

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.3.1 Members

[TIBCBackupService](#) class overview.

### Properties

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BackupFile</a>	Used to set or return the backup file name.
<a href="#">BlockingFactor</a>	Used to set the blocking factor for the tape device as an integer.
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">Database</a>	Used to set or return the database name.
<a href="#">Eof</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">NBackupLevel</a> (inherited from <a href="#">TIBCBBackupRestoreService</a> )	Used to set backup level for nBackup.
<a href="#">NBackupOptions</a> (inherited from <a href="#">TIBCBBackupRestoreService</a> )	Used to set backup options for nBackup.
<a href="#">Options</a>	Used to specify the behaviour of a TIBCBBackupService object.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">UseNBackup</a> (inherited from <a href="#">TIBCBBackupRestoreService</a> )	Used to enable or disable using nBackup service.
<a href="#">Verbose</a>	Used to set or return the backup in the verbose mode.

## Methods

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.

[GetNextLine](#) (inherited from [TIBControlAndQueryService](#)) Returns the next line of data.

[ServiceStart](#) (inherited from [TCustomIBCSERVICE](#)) Starts the service.

## Events

Name	Description
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.3.2 Properties

Properties of the **TIBCBACKUPSERVICE** class.  
For a complete list of the **TIBCBACKUPSERVICE** class members, see the [TIBCBACKUPSERVICE Members](#) topic.

## Public

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Eof</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next line of data.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">NBackupLevel</a> (inherited from <a href="#">TIBCBACKUPRESTORESERVICE</a> )	Used to set backup level for nBackup.
<a href="#">NBackupOptions</a> (inherited from <a href="#">TIBCBACKUPRESTORESERVICE</a> )	Used to set backup options for nBackup.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.
<a href="#">UseNBackup</a> (inherited from <a href="#">TIBCBACKUPRESTORESERVICE</a> )	Used to enable or disable using nBackup service.

## Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BackupFile</a>	Used to set or return the backup file name.



[BlockingFactor](#)

Used to set the blocking factor for the tape device as an integer.

[BufferSize](#) (inherited from [TIBControlAndQueryService](#))

Used to set or return the buffer size.

[Database](#)

Used to set or return the database name.

[LoginPrompt](#) (inherited from [TCustomIBCSERVICE](#))

Used to display a login prompt before attaching to a database.

[OnAttach](#) (inherited from [TCustomIBCSERVICE](#))

Occurs when the database is attached.

[Options](#)

Used to specify the behaviour of a TIBCBACKUPSERVICE object.

[Params](#) (inherited from [TCustomIBCSERVICE](#))

Used to set or return database parameters.

[Protocol](#) (inherited from [TCustomIBCSERVICE](#))

Used to select the network protocol.

[Server](#) (inherited from [TCustomIBCSERVICE](#))

Used to set the name of the server on which the services are to be run.

[Verbose](#)

Used to set or return the backup in the verbose mode.

**See Also**

- [TIBCBACKUPSERVICE Class](#)
- [TIBCBACKUPSERVICE Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.3.2.1 BackupFile Property

Used to set or return the backup file name.

**Class**[TIBCBACKUPSERVICE](#)**Syntax**

```
property BackupFile: TStrings;
```

**Remarks**

Use the BackupFile property to set or return the backup file name.

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.3.2.2 BlockingFactor Property

Used to set the blocking factor for the tape device as an integer.

**Class**

[TIBCBBackupService](#)**Syntax**

```
property BlockingFactor: Integer default 0;
```

**Remarks**

Use the BlockingFactor property to set the blocking factor for the tape device as an integer.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.3.2.3 Database Property

Used to set or return the database name.

**Class**[TIBCBBackupService](#)**Syntax**

```
property Database: string;
```

**Remarks**

Use the Database property to set or return the database name to set properties on.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.3.2.4 Options Property

Used to specify the behaviour of a TIBCBBackupService object.

**Class**[TIBCBBackupService](#)**Syntax**

```
property Options: TIBCBBackupOptions default [];
```

**Remarks**

Set the properties of Options to specify the behaviour of a TIBCBBackupService object.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.3.2.5 Verbose Property

Used to set or return the backup in the verbose mode.

**Class**[TIBCBBackupService](#)**Syntax**

```
property Verbose: Boolean;
```

## Remarks

Use the Verbose property to set or return the backup in the verbose mode.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.4 TIBConfigParams Class

TIBConfigParams holds the configuration information about an InterBase server. For a list of all members of this type, see [TIBConfigParams](#) members.

## Unit

[IBCAAdmin](#)

## Syntax

```
TIBConfigParams = class (System.TObject);
```

## Remarks

TIBConfigParams holds the configuration information about an InterBase server. The TIBConfigParams type stores server configuration settings.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.4.1 Members

[TIBConfigParams](#) class overview.

## Properties

Name	Description
<a href="#">BaseLocation</a>	Used to determine the base location.
<a href="#">LockFileLocation</a>	Used to determine the lock file location.
<a href="#">MessageFileLocation</a>	Used to determine the message file location.
<a href="#">SecurityDatabaseLocation</a>	Used to determine the security database location.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.4.2 Properties

Properties of the **TIBConfigParams** class.  
For a complete list of the **TIBConfigParams** class members, see the [TIBConfigParams Members](#) topic.

## Public

Name	Description
<a href="#">BaseLocation</a>	Used to determine the base location.

[LockFileLocation](#)

Used to determine the lock file location.

[MessageFileLocation](#)

Used to determine the message file location.

[SecurityDatabaseLocation](#)

Used to determine the security database location.

### See Also

- [TIBCCConfigParams Class](#)
  - [TIBCCConfigParams Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.4.2.1 BaseLocation Property

Used to determine the base location.

### Class

[TIBCCConfigParams](#)

### Syntax

```
property BaseLocation: string;
```

### Remarks

Use the BaseLocation property to determine the base location.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.4.2.2 LockFileLocation Property

Used to determine the lock file location.

### Class

[TIBCCConfigParams](#)

### Syntax

```
property LockFileLocation: string;
```

### Remarks

Use the LockFileLocation property to determine the lock file location.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.4.2.3 MessageFileLocation Property

Used to determine the message file location.

### Class

[TIBCCConfigParams](#)

### Syntax

```
property MessageFileLocation: string;
```

**Remarks**

Use the MessageFileLocation to determine the message file location.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.4.2.4 SecurityDatabaseLocation Property

Used to determine the security database location.

**Class**

[TIBCCConfigParams](#)

**Syntax**

```
property SecurityDatabaseLocation: string;
```

**Remarks**

Use the SecurityDatabaseLocation to determine the security database location.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.5 TIBCCConfigService Class

Use a TIBConfigService object to configure database parameters.  
For a list of all members of this type, see [TIBCCConfigService](#) members.

**Unit**

[IBCAAdmin](#)

**Syntax**

```
TIBCCConfigService = class (TCustomIBCSERVICE);
```

**Remarks**

Use a TIBConfigService component to configure database parameters, including page buffers, access mode, and sweep interval.

**Note:** You should install InterBase 6 to use this feature.

**Inheritance Hierarchy**

[TCustomIBCSERVICE](#)

**TIBCCConfigService**

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.5.1 Members

[TIBCCConfigService](#) class overview.

**Properties**

Name	Description
------	-------------

<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">Connection</a>	Used to specify the connection in which the dataset will be executed.
<a href="#">Database</a>	Used to set or return the database name.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">JournalInformation</a>	Holds information about journal system.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">Transaction</a>	Used to determine the transaction under which the query of this dataset executes.

## Methods

Name	Description
<a href="#">ActivateShadow</a>	Activates the database shadow.
<a href="#">AlterJournal</a>	Alters a pre-existing journal system.
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">BringDatabaseOnline</a>	Brings a database online.
<a href="#">CreateJournal</a>	Creates a journal.
<a href="#">CreateJournalArchive</a>	Creates an archive.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">DisableFlush</a>	Disables database flushing.
<a href="#">DropJournal</a>	Drops a journal system.
<a href="#">DropJournalArchive</a>	Drops an archive.
<a href="#">FlushDatabase</a>	Flushes a database.
<a href="#">GetJournalInformation</a>	Retrieves journaling information.
<a href="#">ReclaimMemory</a>	Used to reclaim memory.

<a href="#">ServiceStart</a>	Is not currently used by TIBCCConfigService.
<a href="#">SetAsyncMode</a>	Changes the database write mode to asynchronous (buffered writes).
<a href="#">SetDBSqlDialect</a>	Sets the SQL dialect for the database.
<a href="#">SetFlushInterval</a>	Sets the interval for database flushing
<a href="#">SetGroupCommit</a>	Sets group commit.
<a href="#">SetLingerInterval</a>	Sets the linger interval.
<a href="#">SetPageBuffers</a>	Sets the number of database page buffers.
<a href="#">SetReadOnly</a>	Sets the database to read only.
<a href="#">SetReclaimInterval</a>	Sets the reclaim interval.
<a href="#">SetReserveSpace</a>	Reserves space for versioning.
<a href="#">SetSweepInterval</a>	Sets the sweep interval for the database.
<a href="#">ShutdownDatabase</a>	Shuts down the database.
<a href="#">SweepDatabase</a>	Performs a database sweep.

## Events

Name	Description
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.5.2 Properties

Properties of the **TIBCCConfigService** class.  
For a complete list of the **TIBCCConfigService** class members, see the [TIBCCConfigService Members](#) topic.

## Public

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

## Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">Connection</a>	Used to specify the connection in which the dataset will be executed.
<a href="#">Database</a>	Used to set or return the database name.
<a href="#">JournalInformation</a>	Holds information about journal system.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">Transaction</a>	Used to determine the transaction under which the query of this dataset executes.

### See Also

- [TIBCCConfigService Class](#)
- [TIBCCConfigService Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.5.2.1 Connection Property

Used to specify the connection in which the dataset will be executed.

### Class

[TIBCCConfigService](#)

### Syntax

**property** Connection: [TIBCCConnection](#);

### Remarks

Use the Connection property to specify the connection in which the service will be executed. If connection is not connected, the Open method calls Connection.Connect.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.



## 17.14.1.5.2.2 Database Property

Used to set or return the database name.

**Class**

[TIBCConfigService](#)

**Syntax**

**property** Database: **string**;

**Remarks**

Use the Database property to set or return the database name to set properties on.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.2.3 JournalInformation Property

Holds information about journal system.

**Class**

[TIBCConfigService](#)

**Syntax**

**property** JournalInformation: [TIBCJournalInformation](#);

**Remarks**

The JournalInformation property holds information about a journal system. Gives access to the underlying IBCJournalInformation field.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.2.4 Transaction Property

Used to determine the transaction under which the query of this dataset executes.

**Class**

[TIBCConfigService](#)

**Syntax**

**property** Transaction: [TIBCTransaction](#);

**Remarks**

Use the Transaction property to determine the transaction under which the query of this dataset executes. You can separately set transaction for executing modifying queries with the [TCustomIBCDataset.UpdateTransaction](#) property. By default the Transaction and the UpdateTransaction properties are the same.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3 Methods

Methods of the **TIBCConfigService** class.

For a complete list of the **TIBCConfigService** class members, see the [TIBCConfigService Members](#) topic.

**Public**

Name	Description
<a href="#">ActivateShadow</a>	Activates the database shadow.
<a href="#">AlterJournal</a>	Alters a pre-existing journal system.
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">BringDatabaseOnline</a>	Brings a database online.
<a href="#">CreateJournal</a>	Creates a journal.
<a href="#">CreateJournalArchive</a>	Creates an archive.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">DisableFlush</a>	Disables database flushing.
<a href="#">DropJournal</a>	Drops a journal system.
<a href="#">DropJournalArchive</a>	Drops an archive.
<a href="#">FlushDatabase</a>	Flushes a database.
<a href="#">GetJournalInformation</a>	Retrieves journaling information.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ReclaimMemory</a>	Used to reclaim memory.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a>	Is not currently used by TIBCConfigService.
<a href="#">SetAsyncMode</a>	Changes the database write mode to asynchronous (buffered writes).
<a href="#">SetDBSqlDialect</a>	Sets the SQL dialect for the database.
<a href="#">SetFlushInterval</a>	Sets the interval for database flushing.
<a href="#">SetGroupCommit</a>	Sets group commit.
<a href="#">SetLingerInterval</a>	Sets the linger interval.
<a href="#">SetPageBuffers</a>	Sets the number of database page buffers.
<a href="#">SetReadOnly</a>	Sets the database to read only.
<a href="#">SetReclaimInterval</a>	Sets the reclaim interval.
<a href="#">SetReserveSpace</a>	Reserves space for versioning.
<a href="#">SetSweepInterval</a>	Sets the sweep interval for the database.

[ShutdownDatabase](#)  
[SweepDatabase](#)

Shuts down the database.  
 Performs a database sweep.

## Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.

## See Also

- [TIBCCConfigService Class](#)
- [TIBCCConfigService Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.5.3.1 ActivateShadow Method

Activates the database shadow.

## Class

[TIBCCConfigService](#)

## Syntax

```
procedure ActivateShadow;
```

## Remarks

Call the ActivateShadow method to activate the database shadow. InterBase 6 lets you recover a database in case of disk failure, network failure, or accidental deletion of the database. This recovery method is known as disk shadowing, or simply shadowing. Before you can activate shadowing, you must first create a database shadow, as discussed in 'Shadowing' and 'Creating a shadow' in the InterBase 6 Operations Guide.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.2 AlterJournal Method

Alters a pre-existing journal system.

**Class**

[TIBCConfigService](#)

**Syntax**

```
procedure AlterJournal;
```

**Remarks**

Call the AlterJournal method to alter a pre-existing journal system.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.3 BringDatabaseOnline Method

Brings a database online.

**Class**

[TIBCConfigService](#)

**Syntax**

```
procedure BringDatabaseOnline;
```

**Remarks**

Call the BringDatabaseOnline method to bring a database online.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.4 CreateJournal Method

Creates a journal.

**Class**

[TIBCConfigService](#)

**Syntax**

```
procedure CreateJournal;
```

**Remarks**

Call the CreateJournal method to create a journal based on JournalInformation.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.5 CreateJournalArchive Method

Creates an archive.

**Class**

[TIBCCConfigService](#)

**Syntax**

```
procedure CreateJournalArchive(const Directory: string);
```

**Parameters**

*Directory*

Holds the directory in which the archive is situated.

**Remarks**

Call the CreateJournalArchive method to create an archive.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.6 DisableFlush Method

Disables database flushing.

**Class**

[TIBCCConfigService](#)

**Syntax**

```
procedure DisableFlush;
```

**Remarks**

Call the DisableFlush method to disable database flushing.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.7 DropJournal Method

Drops a journal system.

**Class**

[TIBCCConfigService](#)

**Syntax**

```
procedure DropJournal;
```

**Remarks**

Call the DropJournal method to drop a journal system.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.8 DropJournalArchive Method

Drops an archive.

**Class**

[TIBCCConfigService](#)

**Syntax**

```
procedure DropJournalArchive;
```

**Remarks**

Call the DropJournalArchive to drop an archive.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.9 FlushDatabase Method

Flushes a database.

**Class**

[TIBCCConfigService](#)

**Syntax**

```
procedure FlushDatabase;
```

**Remarks**

Call the FlushDataBase method to flush a database.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.10 GetJournalInformation Method

Retrieves journaling information.

**Class**

[TIBCCConfigService](#)

**Syntax**

```
procedure GetJournalInformation;
```

**Remarks**

Call the GetJournalInformation method to retrieve journaling information for this database and stores it in the [JournalInformation](#) property.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.11 ReclaimMemory Method

Used to reclaim memory.

**Class**

[TIBCConfigService](#)

**Syntax**

```
procedure ReclaimMemory;
```

**Remarks**

Use the ReclaimMemory property to reclaim memory.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.12 ServiceStart Method

Is not currently used by TIBCConfigService.

**Class**

[TIBCConfigService](#)

**Syntax**

```
procedure ServiceStart; override;
```

**Remarks**

Do not use ServiceStart. TIBCConfigService does not currently use this method.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.13 SetAsyncMode Method

Changes the database write mode to asynchronous (buffered writes).

**Class**

[TIBCConfigService](#)

**Syntax**

```
procedure SetAsyncMode(Value: Boolean);
```

**Parameters**

*Value*

True, if the database write mode is changed to asynchronous.

**Remarks**

Set the SetAsyncMode method to True to change the database write mode to asynchronous (buffered writes).

InterBase 6 allows you to write to databases in both synchronous and asynchronous modes. In synchronous mode, the database writes are forced. In asynchronous mode, the database writes are buffered.

For more information, refer to 'Forced writes vs. buffered writes' in the InterBase 6 Operations Guide.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.5.3.14 SetDBSqlDialect Method

Sets the SQL dialect for the database.

#### Class

[TIBConfigService](#)

#### Syntax

```
procedure SetDBSqlDialect(Value: Integer);
```

#### Parameters

*Value*

Holds the appropriate values of SQL dialect (valid values are 1, 2, and 3).

#### Remarks

Call the SetDBSqlDialect method to set the SQL dialect for the database. Valid integer values are 1, 2, and 3.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.5.3.15 SetFlushInterval Method

Sets the interval for database flushing

#### Class

[TIBConfigService](#)

#### Syntax

```
procedure SetFlushInterval(Value: Integer);
```

#### Parameters

*Value*

Holds the interval.

#### Remarks

Call the SetFlushInterval method to set the interval for database flushing.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.5.3.16 SetGroupCommit Method

Sets group commit.

#### Class

[TIBConfigService](#)



## Syntax

```
procedure SetGroupCommit (Value: Boolean);
```

### Parameters

*Value*

## Remarks

Call the SetGroupCommit method to set group commit.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.5.3.17 SetLingerInterval Method

Sets the linger interval.

## Class

[TIBConfigService](#)

## Syntax

```
procedure SetLingerInterval (Value: Integer);
```

### Parameters

*Value*

Holds the linger interval.

## Remarks

Call the SetLingerInterval method to set the linger interval.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.5.3.18 SetPageBuffers Method

Sets the number of database page buffers.

## Class

[TIBConfigService](#)

## Syntax

```
procedure SetPageBuffers (Value: Integer);
```

### Parameters

*Value*

Holds the number of database page buffers.

## Remarks

Call the SetPageBuffers method to set the number of database page buffers.

When a program establishes a connection to a database, InterBase allocates system memory to use as a private buffer. The buffers are used to store accessed database pages to speed performance. The number of buffers assigned determines how many

simultaneous database pages it can have access to in the memory pool. Buffers remain assigned until a program finishes with a database.

For more information on page buffers, refer to 'Setting database cache buffers' in the InterBase 6 Programmer's Guide.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.5.3.19 SetReadOnly Method

Sets the database to read only.

### Class

[TIBCCConfigService](#)

### Syntax

```
procedure SetReadOnly(Value: Boolean);
```

#### Parameters

*Value*

True, if the database is set to read only.

### Remarks

Call the SetReadOnly method to set the database to read only.

Access mode specifies the type of access a transaction has for the table it uses.

There are two possible modes: read-only and read-write.

Read-only specifies that a transaction can read data from a table, but cannot insert, update, or delete table data. Read-write specifies that a transaction can select, insert, update, and delete table data. This is the default setting if none is specified.

**Tip:** You should specify the transaction's access mode even if it is read-write. It makes the application's source code easier to read and debug, because the program's intentions are clearly spelled out.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.5.3.20 SetReclaimInterval Method

Sets the reclaim interval.

### Class

[TIBCCConfigService](#)

### Syntax

```
procedure SetReclaimInterval(Value: Integer);
```

#### Parameters

*Value*

Holds the reclaim interval.

### Remarks

Call the SetReclaimInterval method to set the reclaim interval.

**Note:** You should install InterBase 6 to use this feature.

---

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.5.3.21 SetReserveSpace Method

Reserves space for versioning.

#### Class

[TIBCCConfigService](#)

#### Syntax

```
procedure SetReserveSpace(Value: Boolean);
```

#### Parameters

*Value*

If True, space for versioning is reserved.

#### Remarks

Set SetReserveSpace to true to reserve space on the data page for versioning.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.5.3.22 SetSweepInterval Method

Sets the sweep interval for the database.

#### Class

[TIBCCConfigService](#)

#### Syntax

```
procedure SetSweepInterval(Value: Integer);
```

#### Parameters

*Value*

Holds the sweep interval.

#### Remarks

Call SetSweepInterval set the sweep interval for the database. The sweep interval refers to the number of transactions between database sweeps.

To turn off database sweeps, set the sweep interval to 0.

Sweeping a database is a systematic way of removing outdated records from the database. Periodic sweeping keeps the database from getting too large. However, sweeping can also slow performance. For more information, refer to 'Setting the sweep interval' in the InterBase 6 Operations Guide.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.23 ShutdownDatabase Method

Shuts down the database.

**Class**

[TIBCConfigService](#)

**Syntax**

```
procedure ShutdownDatabase (Options: TIBCSshutdownMode; Wait:  
Integer);
```

**Parameters**

*Options*

*Wait*

Holds the number of seconds to wait before shutting the database.

**Remarks**

Call the ShutdownDatabase method to shut down the database after a specified number of seconds.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.5.3.24 SweepDatabase Method

Performs a database sweep.

**Class**

[TIBCConfigService](#)

**Syntax**

```
procedure SweepDatabase;
```

**Remarks**

Call the SweepDatabase method to perform a database sweep.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

**17.14.1.6 TIBCControlAndQueryService Class**

TIBCControlAndQueryService is the base class from which the log, statistical, validation, security, and backup and restore TIBCSservice components descend. For a list of all members of this type, see [TIBCControlAndQueryService](#) members.

**Unit**

[IBCAdmin](#)

**Syntax**

```
TIBCControlAndQueryService = class (TCustomIBCSservice);
```

**Remarks**

TIBControlAndQueryService is the base class from which the log, statistical, validation, security, and backup and restore TIBService components descend. The service is unstructured because the output is in an unformatted text file.

**Note:** You should install InterBase 6 to use this feature.

## Inheritance Hierarchy

[TCustomIBService](#)

**TIBControlAndQueryService**

© 1997-2013 Devart. All Rights Reserved.

17.14.1.6.1 Members

[TIBControlAndQueryService](#) class overview.

## Properties

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBService</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a>	Used to set or return the buffer size.
<a href="#">Eof</a>	Used to determine whether the end of the file has been reached.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBService</a> )	Used to return the database handle.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBService</a> )	Used to display a login prompt before attaching to a database.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBService</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBService</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBService</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBService</a> )	Used to return and sets SPB parameters.

## Methods

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBService</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBService</a> )	Detaches from the database.
<a href="#">GetNextChunk</a>	Returns the next chunk of data.
<a href="#">GetNextLine</a>	Returns the next line of data.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBService</a> )	Starts the service.

## Events

Name	Description
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.6.2 Properties

Properties of the **TIBCControlAndQueryService** class.

For a complete list of the **TIBCControlAndQueryService** class members, see the [TIBCControlAndQueryService Members](#) topic.

## Public

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Eof</a>	Used to determine whether the end of the file has been reached.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

## Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a>	Used to set or return the buffer size.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.

## See Also

- [TIBCControlAndQueryService Class](#)

- [TIBCCControlAndQueryService Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.6.2.1 BufferSize Property

Used to set or return the buffer size.

#### Class

[TIBCCControlAndQueryService](#)

#### Syntax

```
property BufferSize: Integer;
```

#### Remarks

Use the BufferSize property to set or return the buffer size.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.6.2.2 Eof Property

Used to determine whether the end of the file has been reached.

#### Class

[TIBCCControlAndQueryService](#)

#### Syntax

```
property Eof: Boolean;
```

#### Remarks

Use the Eof property to determine whether the end of the file has been reached.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.6.3 Methods

Methods of the **TIBCCControlAndQueryService** class.

For a complete list of the **TIBCCControlAndQueryService** class members, see the [TIBCCControlAndQueryService Members](#) topic.

#### Public

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">GetNextChunk</a>	Returns the next chunk of data.
<a href="#">GetNextLine</a>	Returns the next line of data.

<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

## Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.

## See Also

- [TIBCCControlAndQueryService Class](#)
- [TIBCCControlAndQueryService Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.6.3.1 GetNextChunk Method

Returns the next chunk of data.

## Class

[TIBCCControlAndQueryService](#)

## Syntax

```
function GetNextChunk: string;
```

## Remarks

Call the GetNextChunk method to get the next chunk of data.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.



## 17.14.1.6.3.2 GetNextLine Method

Returns the next line of data.

**Class**

[TIBCControlAndQueryService](#)

**Syntax**

```
function GetNextLine: string;
```

**Remarks**

Call the GetNextLine method to get the next line of data.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.7 TIBCDatabaseInfo Class

Describes an InterBase database.

For a list of all members of this type, see [TIBCDatabaseInfo](#) members.

**Unit**

[IBCAAdmin](#)

**Syntax**

```
TIBCDatabaseInfo = class(System.TObject);
```

**Remarks**

TIBCDatabaseInfo describes an InterBase database.

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.7.1 Members

[TIBCDatabaseInfo](#) class overview.

**Properties**

Name	Description
<a href="#">DbName</a>	Holds the name of the database.
<a href="#">NoOfAttachments</a>	Holds the number of attachments.

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.7.2 Properties

Properties of the **TIBCDatabaseInfo** class.

For a complete list of the **TIBCDatabaseInfo** class members, see the [TIBCDatabaseInfo Members](#) topic.

**Public**

Name	Description
<a href="#">DbName</a>	Holds the name of the database.
<a href="#">NoOfAttachments</a>	Holds the number of attachments.

**See Also**

- [TIBCDatabaseInfo Class](#)
  - [TIBCDatabaseInfo Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.7.2.1 DbName Property

Holds the name of the database.

**Class**

[TIBCDatabaseInfo](#)

**Syntax**

```
property DbName: TStringDynArray;
```

**Remarks**

The DbName property holds the name of the database.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.7.2.2 NoOfAttachments Property

Holds the number of attachments.

**Class**

[TIBCDatabaseInfo](#)

**Syntax**

```
property NoOfAttachments: integer;
```

**Remarks**

The NoOfAttachments property holds the number of attachments.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.8 TIBCJournalInformation Class

A class used to access the TIBConfigService component properties.  
For a list of all members of this type, see [TIBCJournalInformation](#) members.

**Unit**

[IBCAdmin](#)

**Syntax**

```
TIBCJournalInformation = class (TComponent) ;
```

### Remarks

The TIBCJournalInformation class is used for displaying and setting Journal settings on a database. This class is used to access the TIBConfigService component properties.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.8.1 Members

[TIBCJournalInformation](#) class overview.

### Properties

Name	Description
<a href="#">CheckpointInterval</a>	Used to determine the number of seconds between database checkpoints.
<a href="#">CheckpointLength</a>	Used to determine the number of journal pages to be written before initiating a database checkpoint.
<a href="#">Directory</a>	Holds the name of the directory in which journal is situated.
<a href="#">HasArchive</a>	Turns archiving on.
<a href="#">HasJournal</a>	Turns journaling on.
<a href="#">PageCache</a>	Determines the number of journal buffers that will be allocated.
<a href="#">PageLength</a>	Used to retrieve the page length.
<a href="#">PageSize</a>	Determines the size of a journal page in bytes.
<a href="#">TimestampName</a>	Determines whether or not to append the file creation timestamp to the base journal file name.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.8.2 Properties

Properties of the **TIBCJournalInformation** class.  
For a complete list of the **TIBCJournalInformation** class members, see the [TIBCJournalInformation Members](#) topic.

### Published

Name	Description
------	-------------

[CheckpointInterval](#)

Used to determine the number of seconds between database checkpoints.

[CheckpointLength](#)

Used to determine the number of journal pages to be written before initiating a database checkpoint.

[Directory](#)

Holds the name of the directory in which journal is situated.

[HasArchive](#)

Turns archiving on.

[HasJournal](#)

Turns journaling on.

[PageCache](#)

Determines the number of journal buffers that will be allocated.

[PageLength](#)

Used to retrieve the page length.

[PageSize](#)

Determines the size of a journal page in bytes.

[TimestampName](#)

Determines whether or not to append the file creation timestamp to the base journal file name.

**See Also**

- [TIBCJournalInformation Class](#)
  - [TIBCJournalInformation Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.8.2.1 CheckpointInterval Property

Used to determine the number of seconds between database checkpoints.

**Class**[TIBCJournalInformation](#)**Syntax**

```
property CheckpointInterval: Integer default 0;
```

**Remarks**

Use the CheckpointInterval property to determine the number of seconds between database checkpoints.

**Note:** If both CHECKPOINT LENGTH and CHECKPOINT INTERVAL are specified, whichever event occurs first will initiate a database checkpoint.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.8.2.2 CheckpointLength Property

Used to determine the number of journal pages to be written before initiating a database checkpoint.

**Class**

[TIBCJournalInformation](#)

**Syntax**

```
property CheckpointLength: Integer default 500;
```

**Remarks**

Use the CheckpointInterval property to determine the number of journal pages to be written before initiating a database checkpoint.

**Note:** If both CHECKPOINT LENGTH and CHECKPOINT INTERVAL are specified, whichever event occurs first will initiate a database checkpoint.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.8.2.3 Directory Property

Holds the name of the directory in which journal is situated.

**Class**

[TIBCJournalInformation](#)

**Syntax**

```
property Directory: string;
```

**Remarks**

The Directory property holds the name of the directory in which journal is situated.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.8.2.4 HasArchive Property

Turns archiving on.

**Class**

[TIBCJournalInformation](#)

**Syntax**

```
property HasArchive: Boolean;
```

**Remarks**

Set the HasArchive method to True to turn archiving on.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.8.2.5 HasJournal Property

Turns journaling on.

**Class**

[TIBCJournalInformation](#)

**Syntax**

```
property HasJournal: Boolean;
```

**Remarks**

Set the HasJournal method to True to turn journaling on.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.8.2.6 PageCache Property

Determines the number of journal buffers that will be allocated.

**Class**

[TIBCJournalInformation](#)

**Syntax**

```
property PageCache: Integer default 100;
```

**Remarks**

The PageCache property determines the number of journal buffers that will be allocated. The size of each buffer is the same as the journal page size.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.8.2.7 PageLength Property

Used to retrieve the page length.

**Class**

[TIBCJournalInformation](#)

**Syntax**

```
property PageLength: Integer default 500;
```

**Remarks**

Use the PageLength property to retrieve the page length.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.8.2.8 PageSize Property

Determines the size of a journal page in bytes.

**Class**

[TIBCJournalInformation](#)

## Syntax

```
property PageSize: Integer default 0;
```

## Remarks

The PageSize property determines the size of a journal page in bytes. A journal page size must be at least twice the size of a database page size. If a journal page size of less is specified, it will be rounded up to twice the database page size and a warning will be returned.

The journal page size need not be a power of 2.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.8.2.9 TimestampName Property

Determines whether or not to append the file creation timestamp to the base journal file name.

## Class

[TIBCJournalInformation](#)

## Syntax

```
property TimestampName: Boolean default True;
```

## Remarks

The TimestampName property determines whether or not to append the file creation timestamp to the base journal file name.

If this option is on, the base journal file name will be appended with a timestamp of the form: <YYYY> <MM> <DD>T<hh> <mm> <ss>Z.<sequence-number>.  
journal

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.9 TIBCLicenseInfo Class

Stores information about licensed users.

For a list of all members of this type, see [TIBCLicenseInfo](#) members.

## Unit

[IBCAdmin](#)

## Syntax

```
TIBCLicenseInfo = class(System.TObject);
```

## Remarks

TIBCLicenseInfo stores information about licensed users.

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.9.1 Members

[TIBCLicenseInfo](#) class overview.

**Properties**

Name	Description
<a href="#">Desc</a>	Used to list the description of each licensed user.
<a href="#">Id</a>	Used to list the user Id for each licensed user.
<a href="#">Key</a>	Used to list the license key for each licensed user.
<a href="#">LicensedUsers</a>	Used to determine the number of users.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.9.2 Properties

Properties of the **TIBCLicenseInfo** class.  
For a complete list of the **TIBCLicenseInfo** class members, see the [TIBCLicenseInfo Members](#) topic.

**Public**

Name	Description
<a href="#">Desc</a>	Used to list the description of each licensed user.
<a href="#">Id</a>	Used to list the user Id for each licensed user.
<a href="#">Key</a>	Used to list the license key for each licensed user.
<a href="#">LicensedUsers</a>	Used to determine the number of users.

**See Also**

- [TIBCLicenseInfo Class](#)
  - [TIBCLicenseInfo Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.9.2.1 Desc Property

Used to list the description of each licensed user.

**Class**

[TIBCLicenseInfo](#)

**Syntax**

```
property Desc: TStringDynArray;
```

**Remarks**



---

The Desc property lists the description of each licensed user.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.9.2.2 Id Property

Used to list the user Id for each licensed user.

##### **Class**

[TIBCLicenseInfo](#)

##### **Syntax**

```
property Id: TStringDynArray;
```

##### **Remarks**

The Id property lists the user Id for each licensed user.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.9.2.3 Key Property

Used to list the license key for each licensed user.

##### **Class**

[TIBCLicenseInfo](#)

##### **Syntax**

```
property Key: TStringDynArray;
```

##### **Remarks**

The Key property lists the license key for each licensed user.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.9.2.4 LicensedUsers Property

Used to determine the number of users.

##### **Class**

[TIBCLicenseInfo](#)

##### **Syntax**

```
property LicensedUsers: integer;
```

##### **Remarks**

The LicensedUsers property indicates the number of users (the number of entries in the lists).

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1 TIBCLicenseMaskInfo Class

Indicates the software activation certificate options enabled on the server.  
For a list of all members of this type, see [TIBCLicenseMaskInfo](#) members.

##### Unit

[IBCAdmin](#)

##### Syntax

```
TIBCLicenseMaskInfo = class (System.TObject);
```

##### Remarks

TIBCLicenseMaskInfo indicates the software activation certificate options enabled on the server.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.10.1 Members

[TIBCLicenseMaskInfo](#) class overview.

##### Properties

Name	Description
<a href="#">CapabilityMask</a>	Used to determine a bitmask representing the capabilities currently enabled on the server.
<a href="#">LicenseMask</a>	Used to determine a bitmask representing the software activation certificate options currently enabled on the server.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.10.2 Properties

Properties of the **TIBCLicenseMaskInfo** class.  
For a complete list of the **TIBCLicenseMaskInfo** class members, see the [TIBCLicenseMaskInfo Members](#) topic.

##### Public

Name	Description
<a href="#">CapabilityMask</a>	Used to determine a bitmask representing the capabilities currently enabled on the server.

[LicenseMask](#)

Used to determine a bitmask representing the software activation certificate options currently enabled on the server.

**See Also**

- [TIBCLicenseMaskInfo Class](#)
- [TIBCLicenseMaskInfo Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.10.2.1 CapabilityMask Property

Used to determine a bitmask representing the capabilities currently enabled on the server.

**Class**

[TIBCLicenseMaskInfo](#)

**Syntax**

```
property CapabilityMask: integer;
```

**Remarks**

Use the CapabilityMask to determine a bitmask representing the capabilities currently enabled on the server.

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.10.2.2 LicenseMask Property

Used to determine a bitmask representing the software activation certificate options currently enabled on the server.

**Class**

[TIBCLicenseMaskInfo](#)

**Syntax**

```
property LicenseMask: integer;
```

**Remarks**

Use the Licensemask property to determine a bitmask representing the software activation certificate options currently enabled on the server.

© 1997-2013 Devart. All Rights Reserved.

**17.14.1.1 TIBCLicensingService Class**

TIBCLicensingService configures the licensing parameters  
For a list of all members of this type, see [TIBCLicensingService](#) members.

**Unit**

[IBCAAdmin](#)

## Syntax

```
TIBCLicensingService = class (TCustomIBCSERVICE) ;
```

## Remarks

Use the TIBCLicensingService component to add or remove InterBase software activation certificates.

**Note:** You should install InterBase 6 to use this feature.

## Inheritance Hierarchy

[TCustomIBCSERVICE](#)

**TIBCLicensingService**

© 1997-2013 Devart. All Rights Reserved.

17.14.1.11.1 Members

[TIBCLicensingService](#) class overview.

## Properties

Name	Description
<a href="#">Action</a>	Used to indicate what action the licensing service should take.
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ID</a>	Used to set or return the license identification.
<a href="#">Key</a>	Used to set or return the license key.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.

## Methods

Name	Description
------	-------------

<a href="#">AddLicense</a>	Adds an InterBase software activation certificate.
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">RemoveLicense</a>	Removes an InterBase software activation certificate.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

## Events

Name	Description
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.11.2 Properties

Properties of the **TIBCLicensingService** class.  
For a complete list of the **TIBCLicensingService** class members, see the [TIBCLicensingService Members](#) topic.

## Public

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

## Published

Name	Description
<a href="#">Action</a>	Used to indicate what action the licensing service should take.
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">ID</a>	Used to set or return the license identification.
<a href="#">Key</a>	Used to set or return the license key.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.

<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.

**See Also**

- [TIBCLicensingService Class](#)
  - [TIBCLicensingService Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.11.2.1 Action Property

Used to indicate what action the licensing service should take.

**Class**

[TIBCLicensingService](#)

**Syntax**

```
property Action: TIBCLicensingAction default laAdd;
```

**Remarks**

Use the Action property to indicate what action the licensing service should take. This causes the service to call the [AddLicense](#) or [RemoveLicense](#) method, depending on whether the software activation certificate is added or removed.

**Note:** You should install InterBase 6 to use this feature.

**See Also**

- [AddLicense](#)
  - [RemoveLicense](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.11.2.2 ID Property

Used to set or return the license identification.

**Class**

[TIBCLicensingService](#)

**Syntax**

```
property ID: string;
```

**Remarks**

Use the ID property to set or return the license identification.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.11.2.3 Key Property

Used to set or return the license key.

### Class

[TIBCLicensingService](#)

### Syntax

```
property Key: string;
```

### Remarks

Use the Key property to set or return the license key.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.11.3 Methods

Methods of the **TIBCLicensingService** class.

For a complete list of the **TIBCLicensingService** class members, see the [TIBCLicensingService Members](#) topic.

### Public

Name	Description
<a href="#">AddLicense</a>	Adds an InterBase software activation certificate.
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">RemoveLicense</a>	Removes an InterBase software activation certificate.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

### Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.

<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.

**See Also**

- [TIBCLicensingService Class](#)
  - [TIBCLicensingService Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.11.3.1 AddLicense Method

Adds an InterBase software activation certificate.

**Class**

[TIBCLicensingService](#)

**Syntax**

```
procedure AddLicense;
```

**Remarks**

Call the AddLicense method along to add an InterBase software activation certificate. The [Key](#) and [ID](#) properties should be supplied.

**Note:** You should install InterBase 6 to use this feature.

**See Also**

- [Key](#)
  - [ID](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.11.3.2 RemoveLicense Method

Removes an InterBase software activation certificate.

**Class**

[TIBCLicensingService](#)

**Syntax**

```
procedure RemoveLicense;
```

**Remarks**

Call the RemoveLicense method to remove an InterBase software activation



certificate.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1 TIBCLimboTransactionInfo Class

TIBCLimboTransactionInfo stores information about a limbo transaction.

For a list of all members of this type, see [TIBCLimboTransactionInfo](#) members.

#### Unit

[IBCAAdmin](#)

#### Syntax

```
TIBCLimboTransactionInfo = class (System.TObject);
```

#### Remarks

TIBCLimboTransactionInfo describes limbo transaction. Limbo transactions are usually caused by the failure of a two-phased commit. They can also exist due to system failure or when a single-database transaction is prepared.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.12.1 Members

[TIBCLimboTransactionInfo](#) class overview.

#### Properties

Name	Description
<a href="#">Action</a>	Used to retrieve the action to take for ending the transaction.
<a href="#">Advise</a>	Used to retrieve the suggested action for ending the transaction.
<a href="#">HostSite</a>	Holds the host name for the server where the transaction was started.
<a href="#">ID</a>	Holds the identifier for the limbo transaction.
<a href="#">MultiDatabase</a>	Used to indicate whether the limbo transaction involves multiple databases.
<a href="#">RemoteDatabasePath</a>	Holds the path to the remote server.
<a href="#">RemoteSite</a>	Holds the host name for a remote server involved in the transaction.
<a href="#">State</a>	Used to indicate the current state of the limbo transaction.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.12.2 Properties

Properties of the **TIBCLimboTransactionInfo** class.

For a complete list of the **TIBCLimboTransactionInfo** class members, see the [TIBCLimboTransactionInfo Members](#) topic.

#### Public

Name	Description
<a href="#">Action</a>	Used to retrieve the action to take for ending the transaction.
<a href="#">Advise</a>	Used to retrieve the suggested action for ending the transaction.
<a href="#">HostSite</a>	Holds the host name for the server where the transaction was started.
<a href="#">ID</a>	Holds the identifier for the limbo transaction.
<a href="#">MultiDatabase</a>	Used to indicate whether the limbo transaction involves multiple databases.
<a href="#">RemoteDatabasePath</a>	Holds the path to the remote server.
<a href="#">RemoteSite</a>	Holds the host name for a remote server involved in the transaction.
<a href="#">State</a>	Used to indicate the current state of the limbo transaction.

#### See Also

- [TIBCLimboTransactionInfo Class](#)
  - [TIBCLimboTransactionInfo Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.12.2.1 Action Property

Used to retrieve the action to take for ending the transaction.

#### Class

[TIBCLimboTransactionInfo](#)

#### Syntax

```
property Action: TIBCTransactionAction;
```

#### Remarks

---

Use the Action property to show the action to take for ending the transaction.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.12.2.2 Advise Property

Used to retrieve the suggested action for ending the transaction.

##### **Class**

[TIBCLimboTransactionInfo](#)

##### **Syntax**

```
property Advise: TIBCTransactionAdvise;
```

##### **Remarks**

Use the Advise property to show the suggested action for ending the transaction.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.12.2.3 HostSite Property

Holds the host name for the server where the transaction was started.

##### **Class**

[TIBCLimboTransactionInfo](#)

##### **Syntax**

```
property HostSite: string;
```

##### **Remarks**

The HostSite property holds the host name for the server where the transaction was started.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.12.2.4 ID Property

Holds the identifier for the limbo transaction.

##### **Class**

[TIBCLimboTransactionInfo](#)

##### **Syntax**

```
property ID: integer;
```

##### **Remarks**

The ID property holds the identifier for the limbo transaction.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.12.2.5 MultiDatabase Property

Used to indicate whether the limbo transaction involves multiple databases.

**Class**

[TIBCLimboTransactionInfo](#)

**Syntax**

```
property MultiDatabase: boolean;
```

**Remarks**

Use the MultiDatabase property to indicate whether the limbo transaction involves multiple databases.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.12.2.6 RemoteDatabasePath Property

Holds the path to the remote server.

**Class**

[TIBCLimboTransactionInfo](#)

**Syntax**

```
property RemoteDatabasePath: string;
```

**Remarks**

The RemoteDatabasePath property holds the path to the remote server.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.12.2.7 RemoteSite Property

Holds the host name for a remote server involved in the transaction.

**Class**

[TIBCLimboTransactionInfo](#)

**Syntax**

```
property RemoteSite: string;
```

**Remarks**

The RemoteSite property holds the host name for a remote server involved in the transaction.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.12.2.8 State Property

Used to indicate the current state of the limbo transaction.

**Class**

[TIBCLimboTransactionInfo](#)

## Syntax

```
property State: TIBCTransactionState;
```

## Remarks

Use the State property to indicate the current state of the limbo transaction.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.1:TIBCLogService Class

Returns the contents of the interbase.log file from server.  
For a list of all members of this type, see [TIBCLogService](#) members.

## Unit

[IBCAAdmin](#)

## Syntax

```
TIBCLogService = class(TIBCControlAndQueryService);
```

## Remarks

Use a TIBCLogService object to return the contents of the interbase.log file from server.

## Inheritance Hierarchy

[TCustomIBCSERVICE](#)  
[TIBCControlAndQueryService](#)  
**TIBCLogService**

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.13.1 Members

[TIBCLogService](#) class overview.

## Properties

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a> (inherited from <a href="#">TIBCControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">Eof</a> (inherited from <a href="#">TIBCControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.

<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.

## Methods

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next line of data.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

## Events

Name	Description
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.1 TIBCRestoreService Class

Used to restore a database.

For a list of all members of this type, see [TIBCRestoreService](#) members.

## Unit

[IBCAdmin](#)

## Syntax

```
TIBCRestoreService = class(TIBCBBackupRestoreService);
```

## Remarks

Use a TIBCRestoreService object to restore a database.

**Note:** You should install InterBase 6 to use this feature.

## Inheritance Hierarchy

[TCustomIBCSERVICE](#)  
[TIBControlAndQueryService](#)  
[TIBCBBackupRestoreService](#)  
**TIBCRestoreService**

© 1997-2013 Devart. All Rights Reserved.

17.14.1.14.1 Members

[TIBCRestoreService](#) class overview.

## Properties

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BackupFile</a>	Used to set or return the backup file name.
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">Database</a>	Used to set or return the database name.
<a href="#">Eof</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">NBackupLevel</a> (inherited from <a href="#">TIBCBBackupRestoreService</a> )	Used to set backup level for nBackup.
<a href="#">NBackupOptions</a> (inherited from <a href="#">TIBCBBackupRestoreService</a> )	Used to set backup options for nBackup.
<a href="#">Options</a>	Used to build restore options.
<a href="#">PageBuffers</a>	Used to set or return the page buffer size.
<a href="#">PageSize</a>	Used to set or return the page size.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">UseNBackup</a> (inherited from <a href="#">TIBCBBackupRestoreService</a> )	Used to enable or disable using nBackup service.
<a href="#">Verbose</a>	Used to set or return the restore in the verbose mode.

## Methods

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next line of data.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

## Events

Name	Description
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.14.2 Properties

Properties of the **TIBCRestoreService** class.

For a complete list of the **TIBCRestoreService** class members, see the [TIBCRestoreService Members](#) topic.

## Public

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Eof</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next line of data.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">NBackupLevel</a> (inherited from <a href="#">TIBCBBackupRestoreService</a> )	Used to set backup level for nBackup.
<a href="#">NBackupOptions</a> (inherited from <a href="#">TIBCBBackupRestoreService</a> )	Used to set backup options for nBackup.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.
<a href="#">UseNBackup</a> (inherited from <a href="#">TIBCBBackupRestoreService</a> )	Used to enable or disable using nBackup service.

## Published



Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BackupFile</a>	Used to set or return the backup file name.
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">Database</a>	Used to set or return the database name.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Options</a>	Used to build restore options.
<a href="#">PageBuffers</a>	Used to set or return the page buffer size.
<a href="#">PageSize</a>	Used to set or return the page size.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">Verbose</a>	Used to set or return the restore in the verbose mode.

### See Also

- [TIBCRestoreService Class](#)
- [TIBCRestoreService Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.14.2.1 BackupFile Property

Used to set or return the backup file name.

### Class

[TIBCRestoreService](#)

### Syntax

```
property BackupFile: TStrings;
```

### Remarks

Use the BackupFile property to set or return the backup file name.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.14.2.2 Database Property

Used to set or return the database name.

##### **Class**

[TIBCRestoreService](#)

##### **Syntax**

```
property Database: TStrings;
```

##### **Remarks**

Use the Database property to set or return the database name to set properties on.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.14.2.3 Options Property

Used to build restore options.

##### **Class**

[TIBCRestoreService](#)

##### **Syntax**

```
property Options: TIBCRestoreOptions default [roCreateNewDB];
```

##### **Remarks**

Use the Options property to build restore options of type TIBCRestoreOption into application.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.14.2.4 PageBuffers Property

Used to set or return the page buffer size.

##### **Class**

[TIBCRestoreService](#)

##### **Syntax**

```
property PageBuffers: Integer default 0;
```

##### **Remarks**

Use the PageBuffers property to set or return the page buffer size in kilobytes.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.14.2.5 PageSize Property

Used to set or return the page size.

**Class**

[TIBCRestoreService](#)

**Syntax**

```
property PageSize: Integer default 4096;
```

**Remarks**

Use the PageSize property to set or return the page size in kilobytes.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.14.2.6 Verbose Property

Used to set or return the restore in the verbose mode.

**Class**

[TIBCRestoreService](#)

**Syntax**

```
property Verbose: Boolean;
```

**Remarks**

Use the Verbose property to set or return the restore in the verbose mode.

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.14.1 TIBCSecurityService Class

Used to manage user access to the InterBase server.

For a list of all members of this type, see [TIBCSecurityService](#) members.

**Unit**

[IBCAdmin](#)

**Syntax**

```
TIBCSecurityService = class(TIBControlAndQueryService);
```

**Remarks**

Use a TIBCSecurityService object to manage user access to the InterBase server.

**Note:** You should install InterBase 6 to use this feature.

**Inheritance Hierarchy**

[TCustomIBCSERVICE](#)

[TIBControlAndQueryService](#)

**TIBCSecurityService**

© 1997-2013 Devart. All Rights Reserved.

17.14.1.15.1 Members

[TIBCSecurityService](#) class overview.

## Properties

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">Eof</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">SecurityAction</a>	Used to determine the type of the operation that will be performed by InterBase Security Service.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">UserDatabase</a>	Used to add, modify, or delete a user in the specified database.
<a href="#">UserInfo</a>	Used to return the user's information.
<a href="#">UserInfos</a>	Used to get the user's information.
<a href="#">UserInfosCount</a>	Used to get the number of returned records.

## Methods

Name	Description
<a href="#">AddUser</a>	Adds a user to the user table.
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.

<a href="#">DeleteUser</a>	Used to delete a user from the user table.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">DisplayUser</a>	Fetches user information and dumps it into the TIBCUserInfo record
<a href="#">DisplayUsers</a>	Fetches all valid user information.
<a href="#">EnableEUA</a>	Enables user authentication.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Returns the next line of data.
<a href="#">ModifyUser</a>	Modifies user's information.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.
<a href="#">SuspendEUA</a>	Suspends embedded user authentication.

## Events

Name	Description
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.15.2 Properties

Properties of the **TIBCSecurityService** class.  
For a complete list of the **TIBCSecurityService** class members, see the [TIBCSecurityService Members](#) topic.

## Public

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Eof</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Returns the next line of data.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.

<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.
<a href="#">UserInfos</a>	Used to get the user's information.
<a href="#">UserInfosCount</a>	Used to get the number of returned records.

## Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">SecurityAction</a>	Used to determine the type of the operation that will be performed by InterBase Security Service.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">UserDatabase</a>	Used to add, modify, or delete a user in the specified database.
<a href="#">UserInfo</a>	Used to return the user's information.

## See Also

- [TIBCSecurityService Class](#)
- [TIBCSecurityService Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.15.2.1 SecurityAction Property

Used to determine the type of the operation that will be performed by InterBase Security Service.

## Class

[TIBCSecurityService](#)

## Syntax

```
property SecurityAction: TIBCSecurityAction default saAddUser;
```

### Remarks

The SecurityAction property determines the type of the operation that will be performed by InterBase Security Service.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.15.2.2 UserDatabase Property

Used to add, modify, or delete a user in the specified database.

### Class

[TIBCSecurityService](#)

### Syntax

```
property UserDatabase: string;
```

### Remarks

Set the UserDatabase property to add, modify, or delete a user in the specified database.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.15.2.3 UserInfo Property

Used to return the user's information.

### Class

[TIBCSecurityService](#)

### Syntax

```
property UserInfo: TIBCUUserInfo;
```

### Remarks

Fill in the UserInfo property to add, delete, or modify a user.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.15.2.4 UserInfos Property(Indexer)

Used to get the user's information.

### Class

[TIBCSecurityService](#)

### Syntax

```
property UserInfos[Index: Integer]: TIBCUUserInfo;
```

## Parameters

### *Index*

Holds the index of the user to get information on.

## Remarks

Use the UserInfos property to return the user's information as a TIBCUserInfo record, which contains the user name, groupID, UserID, and the user's first, middle, and last names. Specify the index of the required user in the Index parameter.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.15.2.5 UserInfosCount Property

Used to get the number of returned records.

## Class

[TIBCSecurityService](#)

## Syntax

```
property UserInfosCount: Integer;
```

## Remarks

Use the UserInfosCount to get the number of returned TIBCUserInfo records.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.15.3 Methods

Methods of the **TIBCSecurityService** class.

For a complete list of the **TIBCSecurityService** class members, see the [TIBCSecurityService Members](#) topic.

## Public

Name	Description
<a href="#">AddUser</a>	Adds a user to the user table.
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">DeleteUser</a>	Used to delete a user from the user table.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">DisplayUser</a>	Fetches user information and dumps it into the TIBCUserInfo record
<a href="#">DisplayUsers</a>	Fetches all valid user information.
<a href="#">EnableEUA</a>	Enables user authentication.
<a href="#">Eof</a> (inherited from <a href="#">TIBCControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.



<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next line of data.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ModifyUser</a>	Modifies user's information.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.
<a href="#">SuspendEUA</a>	Suspends embedded user authentication.

## Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.

## See Also

- [TIBSecurityService Class](#)
- [TIBSecurityService Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.15.3.1 AddUser Method

Adds a user to the user table.

## Class

[TIBSecurityService](#)

## Syntax

```
procedure AddUser;
```

**Remarks**

Call the AddUser method to add a user to the user table.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.15.3.2 DeleteUser Method

Used to delete a user from the user table.

**Class**

[TIBCSecurityService](#)

**Syntax**

```
procedure DeleteUser;
```

**Remarks**

Call the DeleteUser method to delete a user from the user table.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.15.3.3 DisplayUser Method

Fetches user information and dumps it into the TIBCUUserInfo record

**Class**

[TIBCSecurityService](#)

**Syntax**

```
procedure DisplayUser(const UserName: string);
```

**Parameters**

*UserName*

Holds the user name.

**Remarks**

Call the DisplayUser method to fetch user information and dump it into the TIBCUUserInfo record, which can then be returned by the [UserInfos](#) property.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.15.3.4 DisplayUsers Method

Fetches all valid user information.

**Class**

[TIBCSecurityService](#)

**Syntax**

---

```
procedure DisplayUsers;
```

**Remarks**

Call the DisplayUsers method to fetch all valid user information, which can then be returned by the [UserInfo](#) property.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.15.3.5 EnableEUA Method

Enables user authentication.

**Class**

[TIBCSecurityService](#)

**Syntax**

```
procedure EnableEUA(Value: Boolean);
```

**Parameters**

*Value*

True if EUA is enabled.

**Remarks**

Call the EnableEUA method to enable embedded user authentication.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.15.3.6 ModifyUser Method

Modifies user's information.

**Class**

[TIBCSecurityService](#)

**Syntax**

```
procedure ModifyUser;
```

**Remarks**

Call ModifyUser to modify user's information.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.15.3.7 SuspendEUA Method

Suspends embedded user authentication.

**Class**

[TIBCSecurityService](#)

**Syntax**

```
procedure SuspendEUA(Value: Boolean);
```

**Parameters***Value*

True, if embedded user authentication is suspended.

**Remarks**

Call the SuspendEUA method to suspend embedded user authentication.

**Note:** You should install InterBase 6 to use this feature.

---

*© 1997-2013 Devart. All Rights Reserved.***17.14.1.1 TIBCServerProperties Class**

A class that returns database server information.

For a list of all members of this type, see [TIBCServerProperties](#) members.**Unit**[IBCAAdmin](#)**Syntax**

```
TIBCServerProperties = class (TCustomIBCSERVICE);
```

**Remarks**

The TIBCServerProperties class returns database server information, including configuration parameters, and also version and license information.

**Note:** You must install InterBase 6 to use this feature.**Inheritance Hierarchy**[TCustomIBCSERVICE](#)**TIBCServerProperties**

---

*© 1997-2013 Devart. All Rights Reserved.***17.14.1.16.1 Members**[TIBCServerProperties](#) class overview.**Properties**

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">AliasCount</a>	Used to get the alias count.
<a href="#">ConfigParams</a>	Used to return server configuration parameters.
<a href="#">DatabaseInfo</a>	Used to get database information.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.

[LicenseInfo](#)

Used to get licensing information.

[LicenseMaskInfo](#)

Used to return licensing information as a TIBCLicenseMaskInfo record, which includes the license mask and capability mask.

[LoginPrompt](#) (inherited from [TCustomIBCSERVICE](#))

Used to display a login prompt before attaching to a database.

[Params](#) (inherited from [TCustomIBCSERVICE](#))

Used to set or return database parameters.

[Protocol](#) (inherited from [TCustomIBCSERVICE](#))

Used to select the network protocol.

[Server](#) (inherited from [TCustomIBCSERVICE](#))

Used to set the name of the server on which the services are to be run.

[ServiceParamBySPB](#) (inherited from [TCustomIBCSERVICE](#))

Used to return and sets SPB parameters.

[VersionInfo](#)

Used to get the server configuration parameters.

## Methods

### Name

### Description

[AddAlias](#)

Adds database alias.

[Attach](#) (inherited from [TCustomIBCSERVICE](#))

Attaches to the database.

[DeleteAlias](#)

Deletes database alias.

[Detach](#) (inherited from [TCustomIBCSERVICE](#))

Detaches from the database.

[Fetch](#)

Gets information from the server.

[FetchAliasInfo](#)

Returns database alias information.

[FetchConfigParams](#)

Returns database configuration parameters.

[FetchDatabaseInfo](#)

Returns database information.

[FetchLicenseInfo](#)

Returns database license information.

[FetchLicenseMaskInfo](#)

Returns license mask information.

[FetchVersionInfo](#)

Returns the database version information.

[ServiceStart](#) (inherited from [TCustomIBCSERVICE](#))

Starts the service.

## Events

### Name

### Description

[OnAttach](#) (inherited from [TCustomIBCSERVICE](#)) Occurs when the database is attached.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.2 Properties

Properties of the **TIBCSERVERPROPERTIES** class.  
For a complete list of the **TIBCSERVERPROPERTIES** class members, see the [TIBCSERVERPROPERTIES MEMBERS](#) topic.

### Public

Name	Description
<a href="#">AliasCount</a>	Used to get the alias count.
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">ConfigParams</a>	Used to return server configuration parameters.
<a href="#">DatabaseInfo</a>	Used to get database information.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">LicenseInfo</a>	Used to get licensing information.
<a href="#">LicenseMaskInfo</a>	Used to return licensing information as a TIBCLicenseMaskInfo record, which includes the license mask and capability mask.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.
<a href="#">VersionInfo</a>	Used to get the server configuration parameters.

### Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.

[Server](#) (inherited from [TCustomIBCSERVICE](#))

Used to set the name of the server on which the services are to be run.

### See Also

- [TIBCServerProperties Class](#)
- [TIBCServerProperties Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.2.1 AliasCount Property

Used to get the alias count.

### Class

[TIBCServerProperties](#)

### Syntax

```
property AliasCount: Integer;
```

### Remarks

Use the AliasCount property to obtain the alias count.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.2.2 ConfigParams Property

Used to return server configuration parameters.

### Class

[TIBCServerProperties](#)

### Syntax

```
property ConfigParams: TIBConfigParams;
```

### Remarks

Use the ConfigParams property to return the server configuration parameters as a TIBConfigParams record, which includes the configuration file data, base location, lock file location, message file location, and the security database location.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.2.3 DatabaseInfo Property

Used to get database information.

### Class

[TIBCServerProperties](#)

### Syntax

**property** DatabaseInfo: [TIBCDatabaseInfo](#);

### Remarks

Use the DatabaseInfo property to obtain database information, which includes the database name, number of attachments, and number of databases.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.2.4 LicenseInfo Property

Used to get licensing information.

### Class

[TIBCServerProperties](#)

### Syntax

**property** LicenseInfo: [TIBCLicenseInfo](#);

### Remarks

Use the LicenseInfo property to return licensing information as a TIBCLicenseInfo record, which includes the key, ID, and number of licensed users.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.2.5 LicenseMaskInfo Property

Used to return licensing information as a TIBCLicenseMaskInfo record, which includes the license mask and capability mask.

### Class

[TIBCServerProperties](#)

### Syntax

**property** LicenseMaskInfo: [TIBCLicenseMaskInfo](#);

### Remarks

Use the LicenseMaskInfo property to return licensing information as a TIBCLicenseMaskInfo record, which includes the license mask and capability mask. A license mask is a bitmask representing the software activation certificate options currently enabled on the server. A capability mask is a bitmask representing the capabilities currently enabled on the server.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.2.6 VersionInfo Property

Used to get the server configuration parameters.

### Class



## [TIBCServerProperties](#)

### Syntax

**property** VersionInfo: [TIBCVersionInfo](#);

### Remarks

Use the VersionInfo property to obtain the server configuration parameters as a TIBCVersionInfo type, which includes the service version, and server version and implementation.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.3 Methods

Methods of the **TIBCServerProperties** class.

For a complete list of the **TIBCServerProperties** class members, see the [TIBCServerProperties Members](#) topic.

### Public

Name	Description
<a href="#">AddAlias</a>	Adds database alias.
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">DeleteAlias</a>	Deletes database alias.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Fetch</a>	Gets information from the server.
<a href="#">FetchAliasInfo</a>	Returns database alias information.
<a href="#">FetchConfigParams</a>	Returns database configuration parameters.
<a href="#">FetchDatabaseInfo</a>	Returns database information.
<a href="#">FetchLicenseInfo</a>	Returns database license information.
<a href="#">FetchLicenseMaskInfo</a>	Returns license mask information.
<a href="#">FetchVersionInfo</a>	Returns the database version information.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

### Published

Name	Description
------	-------------

<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.

### See Also

- [TIBCSERVERPROPERTIES Class](#)
- [TIBCSERVERPROPERTIES Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.3.1 AddAlias Method

Adds database alias.

### Class

[TIBCSERVERPROPERTIES](#)

### Syntax

```
procedure AddAlias(const Alias: string; const DBPath: string);
```

### Parameters

*Alias*

Holds the database alias.

*DBPath*

### Remarks

Call the AddAlias method to add database alias.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.3.2 DeleteAlias Method

Deletes database alias.

### Class

[TIBCSERVERPROPERTIES](#)

### Syntax

```
procedure DeleteAlias(const Alias: string);
```

### Parameters

*Alias*

Holds database alias.

### Remarks

Call the DeleteAlias method to delete database alias.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.3.3 Fetch Method

Gets information from the server.

### Class

[TIBCServerProperties](#)

### Syntax

```
procedure Fetch;
```

### Remarks

Call the Fetch method to get information from the server.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.3.4 FetchAliasInfo Method

Returns database alias information.

### Class

[TIBCServerProperties](#)

### Syntax

```
procedure FetchAliasInfo;
```

### Remarks

Call the FetchAliasInfo method to return database alias information.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.3.5 FetchConfigParams Method

Returns database configuration parameters.

### Class

[TIBCServerProperties](#)

### Syntax

```
procedure FetchConfigParams;
```

**Remarks**

Call the FetchConfigParams method to return database configuration parameters.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.16.3.6 FetchDatabaseInfo Method

Returns database information.

**Class**

[TIBCServerProperties](#)

**Syntax**

```
procedure FetchDatabaseInfo;
```

**Remarks**

Call the FetchDatabaseInfo method to return database information.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.16.3.7 FetchLicenseInfo Method

Returns database license information.

**Class**

[TIBCServerProperties](#)

**Syntax**

```
procedure FetchLicenseInfo;
```

**Remarks**

Call the FetchLicenseInfo method to return database license information.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.16.3.8 FetchLicenseMaskInfo Method

Returns license mask information.

**Class**

[TIBCServerProperties](#)

**Syntax**

```
procedure FetchLicenseMaskInfo;
```

**Remarks**

Call the FetchLicenseMaskInfo method to return database license mask information.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.16.3.9 FetchVersionInfo Method

Returns the database version information.

### Class

[TIBCServerProperties](#)

### Syntax

```
procedure FetchVersionInfo;
```

### Remarks

Call the FetchVersionInfo method to return the database version information.

**Note:** You should install InterBase 6 to use this feature.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.1 TIBCSStatisticalService Class

TIBCSStatisticalService is used to view database statistics.

For a list of all members of this type, see [TIBCSStatisticalService](#) members.

### Unit

[IBCAAdmin](#)

### Syntax

```
TIBCSStatisticalService = class(TIBCControlAndQueryService);
```

### Remarks

Use a TIBCSStatisticalService object to view database statistics.

TIBCSStatisticalService contains many properties and methods to allow you to build a statistical component into your application. Only the SYSDBA user or owner of the database will be able to run this service.

### Inheritance Hierarchy

[TCustomIBCSERVICE](#)

[TIBCControlAndQueryService](#)

**TIBCSStatisticalService**

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.17.1 Members

[TIBCSStatisticalService](#) class overview.

### Properties

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).

<a href="#">BufferSize</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">Database</a>	Used to set or return the database name.
<a href="#">Eof</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">Options</a>	Used to specify the behaviour of a TIBCSStatisticalService object.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">TableNames</a>	Used to specify the name of the table to request statistics for.

## Methods

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Returns the next line of data.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

## Events

Name	Description
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.

## 17.14.1.17.2 Properties

Properties of the **TIBCStatisticalService** class.

For a complete list of the **TIBCStatisticalService** class members, see the [TIBCStatisticalService Members](#) topic.

**Public**

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Eof</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next line of data.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

**Published**

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">Database</a>	Used to set or return the database name.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Options</a>	Used to specify the behaviour of a TIBCStatisticalService object.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.

[TableNames](#)

Used to specify the name of the table to request statistics for.

**See Also**

- [TIBCStatisticalService Class](#)
  - [TIBCStatisticalService Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.17.2.1 Database Property

Used to set or return the database name.

**Class**

[TIBCStatisticalService](#)

**Syntax**

```
property Database: string;
```

**Remarks**

Use the Database property to set or return the database name to set properties on.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.17.2.2 Options Property

Used to specify the behaviour of a TIBCStatisticalService object.

**Class**

[TIBCStatisticalService](#)

**Syntax**

```
property Options: TIBCStatOptions;
```

**Remarks**

Use the Options property of TIBCStatisticalService to request database statistics. These options are incremental; that is, setting DbLog to True also returns HeaderPages statistics, setting IndexPages to True returns also returns DbLog and HeaderPages statistics, and so forth.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.17.2.3 TableNames Property

Used to specify the name of the table to request statistics for.

**Class**

[TIBCStatisticalService](#)

**Syntax**

```
property TableNames: string;
```



## Remarks

Use the TableName property to specify the name of the table for which statistics should be requested.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.1 TIBCTraceService Class

This component is used for working with trace service added in Firebird 2.5. For a list of all members of this type, see [TIBCTraceService](#) members.

## Unit

[IBCAdmin](#)

## Syntax

```
TIBCTraceService = class (TIBCCControlAndQueryService) ;
```

## Remarks

The TIBCTraceService component is used to work with trace service added in Firebird 2.5.

Use the TIBCTraceService component to work with trace service. Trace enables various events performed inside the engine, such as statement execution, connections, disconnections, etc., to be logged and collated for real-time analysis of the corresponding performance characteristics.

Trace takes place in the context of a trace session. Each trace session has its own configuration, state and output. When a user application starts a trace session, it sets SessionName (optional) and the session configuration (Config) (mandatory).

**Note:** trace service is supported starting with Firebird 2.5.

## Inheritance Hierarchy

[TCustomIBCSERVICE](#)  
[TIBCCControlAndQueryService](#)  
**TIBCTraceService**

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.18.1 Members

[TIBCTraceService](#) class overview.

## Properties

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">Config</a>	Used to set the configuration of a trace session.

<a href="#">Eof</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">SessionName</a>	Used to set session name for a trace session.

## Methods

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next line of data.
<a href="#">ListTraceSessions</a>	Returns information about all trace sessions.
<a href="#">ResumeTrace</a>	Resumes a trace session.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.
<a href="#">StartTrace</a>	Starts the trace session.
<a href="#">StopTrace</a>	Stops a trace session.
<a href="#">SuspendTrace</a>	Suspends a trace session.

## Events

Name	Description
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.

© 1997-2013 Devart. All Rights Reserved.

17.14.1.18.2 Properties

Properties of the **TIBCTraceService** class.  
For a complete list of the **TIBCTraceService** class members, see the

[TIBCTraceService Members](#) topic.

## Public

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Eof</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next line of data.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

## Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">Config</a>	Used to set the configuration of a trace session.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">SessionName</a>	Used to set session name for a trace session.

## See Also

- [TIBCTraceService Class](#)
- [TIBCTraceService Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.18.2.1 Config Property

Used to set the configuration of a trace session.

**Class**

[TIBCTraceService](#)

**Syntax**

```
property Config: TStrings;
```

**Remarks**

Use the Config property to set the configuration of a trace session.

The session configuration is a text conforming to the rules and syntax modelled in the fbtrace.conf template that is in Firebird's root directory, apart from the lines relating to placement of the output.

For example, the following configuration is used to trace prepare, free and execution of all statements in the mydatabase.fdb database:

```
<database mydatabase.fdb>
  enabled true
  log_statement_prepare true
  log_statement_free true
  log_statement_start true
  log_statement_finish true
  time_threshold 0
</database>
```

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.18.2.2 SessionName Property

Used to set session name for a trace session.

**Class**

[TIBCTraceService](#)

**Syntax**

```
property SessionName: string;
```

**Remarks**

Set the SessionName property to distinguish your session in the list of trace sessions.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.18.3 Methods

Methods of the **TIBCTraceService** class.

For a complete list of the **TIBCTraceService** class members, see the [TIBCTraceService Members](#) topic.

**Public**

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.

<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Eof</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next line of data.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ListTraceSessions</a>	Returns information about all trace sessions.
<a href="#">ResumeTrace</a>	Resumes a trace session.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.
<a href="#">StartTrace</a>	Starts the trace session.
<a href="#">StopTrace</a>	Stops a trace session.
<a href="#">SuspendTrace</a>	Suspends a trace session.

## Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.

## See Also

- [TIBCTraceService Class](#)
- [TIBCTraceService Class Members](#)

## 17.14.1.18.3.1 ListTraceSessions Method

Returns information about all trace sessions.

**Class**

[TIBCTraceService](#)

**Syntax**

```
procedure ListTraceSessions;
```

**Remarks**

Call the ListTraceSessions method to get information about all trace sessions. You can read this information from the service output using the GetNextChunk or GetNextLine methods. For each session the service returns a text message in the following format:

- Session ID: <number>
  - name: <string>. Prints the trace session name if it is not empty
  - user: <string>. Prints the user name of the user that created the trace session
  - date: YYYY-MM-DD HH:NN:SS, start date and time of the user session
  - flags: <string>
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.18.3.2 ResumeTrace Method

Resumes a trace session.

**Class**

[TIBCTraceService](#)

**Syntax**

```
procedure ResumeTrace(SessionID: integer);
```

**Parameters**

*SessionID*

Holds a session ID.

**Remarks**

Call the ResumeTrace method to resume the trace session with the specified ID. You can find the ID of the current trace session in the first lines of its output. Use ListTraceSessions to get ID for all trace sessions.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.18.3.3 StartTrace Method

Starts the trace session.

**Class**

[TIBCTraceService](#)

**Syntax**

```
procedure StartTrace;
```

### Remarks

Call the StartTrace method to start the trace session. After the session is started you can read its output using the GetNextChunk or GetNextLine methods.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.18.3.4 StopTrace Method

Stops a trace session.

### Class

[TIBCTraceService](#)

### Syntax

```
procedure StopTrace(SessionID: integer);
```

#### Parameters

*SessionID*

Holds a session ID.

### Remarks

Call the StopTrace method to stop the trace session with the specified ID. You can find the ID of the current trace session in the first lines of its output. Use ListTraceSessions to get ID for all trace sessions.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.18.3.5 SuspendTrace Method

Suspends a trace session.

### Class

[TIBCTraceService](#)

### Syntax

```
procedure SuspendTrace(SessionID: integer);
```

#### Parameters

*SessionID*

Holds a session ID.

### Remarks

Call the SuspendTrace method to suspend the trace session with the specified ID. You can find ID of the current trace session in the first lines of its output. Use ListTraceSessions to get ID for all trace sessions.

© 1997-2013 Devart. All Rights Reserved.

**17.14.1.1 TIBUserInfo Class**

TIBUserInfo stores information about an InterBase user for the security service. For a list of all members of this type, see [TIBUserInfo](#) members.

**Unit**

[IBCAAdmin](#)

**Syntax**

```
TIBUserInfo = class (System.TObject) ;
```

**Remarks**

TIBUserInfo is the user descriptor that the InterBase security service uses to describe a single user.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.19.1 Members

[TIBUserInfo](#) class overview.

**Properties**

Name	Description
<a href="#">ActiveUser</a>	Used to determine if the user is active.
<a href="#">DefaultRole</a>	Holds the user default role.
<a href="#">Description</a>	Holds the description.
<a href="#">FirstName</a>	Holds the user's first name.
<a href="#">GroupID</a>	Holds the user's group ID.
<a href="#">GroupName</a>	Holds the name of the group.
<a href="#">LastName</a>	Holds the user's Last name.
<a href="#">MiddleName</a>	Holds the user's middle name.
<a href="#">Password</a>	Holds the user's password.
<a href="#">SQLRole</a>	Holds an optional role to use when attaching to the security database.
<a href="#">SystemUserName</a>	Holds the system user name.
<a href="#">UserID</a>	Holds the user's individual ID.
<a href="#">UserName</a>	Holds the user's login name.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.19.2 Properties

Properties of the **TIBUserInfo** class.  
For a complete list of the **TIBUserInfo** class members, see the [TIBUserInfo](#)



[Members](#) topic.

## Public

Name	Description
<a href="#">ActiveUser</a>	Used to determine if the user is active.
<a href="#">DefaultRole</a>	Holds the user default role.
<a href="#">Description</a>	Holds the description.
<a href="#">FirstName</a>	Holds the user's first name.
<a href="#">GroupID</a>	Holds the user's group ID.
<a href="#">GroupName</a>	Holds the name of the group.
<a href="#">LastName</a>	Holds the user's Last name.
<a href="#">MiddleName</a>	Holds the user's middle name.
<a href="#">Password</a>	Holds the user's password.
<a href="#">SQLRole</a>	Holds an optional role to use when attaching to the security database.
<a href="#">SystemUserName</a>	Holds the system user name.
<a href="#">UserID</a>	Holds the user's individual ID.
<a href="#">UserName</a>	Holds the user's login name.

## See Also

- [TIBCUserInfo Class](#)
- [TIBCUserInfo Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

17.14.1.19.2.1 ActiveUser Property

Used to determine if the user is active.

## Class

[TIBCUserInfo](#)

## Syntax

```
property ActiveUser: Boolean;
```

## Remarks

Use the ActiveUser property to find out if the user is active.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.19.2.2 DefaultRole Property

Holds the user default role.

**Class**

[TIBCUserInfo](#)

**Syntax**

```
property DefaultRole: string;
```

**Remarks**

The DefaultRole property holds the user default role.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.19.2.3 Description Property

Holds the description.

**Class**

[TIBCUserInfo](#)

**Syntax**

```
property Description: string;
```

**Remarks**

The Description property holds the description.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.19.2.4 FirstName Property

Holds the user's first name.

**Class**

[TIBCUserInfo](#)

**Syntax**

```
property FirstName: string;
```

**Remarks**

The FirstName property holds the user's first name.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.19.2.5 GroupID Property

Holds the user's group ID.

**Class**

[TIBCUserInfo](#)

**Syntax**

```
property GroupID: Integer;
```

**Remarks**

The GroupID property holds the user's group ID.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.19.2.6 GroupName Property

Holds the name of the group.

**Class**

[TIBCUserInfo](#)

**Syntax**

```
property GroupName: string;
```

**Remarks**

The GroupName property holds the name of the group.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.19.2.7 LastName Property

Holds the user's Last name.

**Class**

[TIBCUserInfo](#)

**Syntax**

```
property LastName: string;
```

**Remarks**

The LastName property holds the user's Last name.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.19.2.8 MiddleName Property

Holds the user's middle name.

**Class**

[TIBCUserInfo](#)

**Syntax**

```
property MiddleName: string;
```

**Remarks**

The MiddleName property holds the user's middle name.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.19.2.9 Password Property

Holds the user's password.

##### **Class**

[TIBCUserInfo](#)

##### **Syntax**

```
property Password: string;
```

##### **Remarks**

The Password property holds the user's password. The password can be maximum 31 characters, only first 8 characters are significant

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.19.2.10 SQLRole Property

Holds an optional role to use when attaching to the security database.

##### **Class**

[TIBCUserInfo](#)

##### **Syntax**

```
property SQLRole: string;
```

##### **Remarks**

The SQLRole property is an optional role to use when attaching to the security database.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.19.2.11 SystemUserName Property

Holds the system user name.

##### **Class**

[TIBCUserInfo](#)

##### **Syntax**

```
property SystemUserName: string;
```

##### **Remarks**

The SystemUserName property holds the system user name.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.19.2.12 UserID Property

Holds the user's individual ID.

**Class**

[TIBCUserInfo](#)

**Syntax**

```
property UserID: Integer;
```

**Remarks**

The UserID property holds the the user's individual ID.

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.19.2.13 UserName Property

Holds the user's login name.

**Class**

[TIBCUserInfo](#)

**Syntax**

```
property UserName: string;
```

**Remarks**

The UserName property holds the user's login name.

© 1997-2013 Devart. All Rights Reserved.

**17.14.1.2TIBCValidationService Class**

Used to validate a database and reconcile database transactions.  
For a list of all members of this type, see [TIBCValidationService](#) members.

**Unit**

[IBCAdmin](#)

**Syntax**

```
TIBCValidationService = class(TIBCControlAndQueryService);
```

**Remarks**

Use a TIBCValidationService object to validate a database and reconcile database transactions.

**Note:** You must install InterBase 6 to use this feature.

**Inheritance Hierarchy**

[TCustomIBCSERVICE](#)

[TIBCControlAndQueryService](#)

**TIBCValidationService**

© 1997-2013 Devart. All Rights Reserved.

17.14.1.20.1 Members

[TIBCVValidationService](#) class overview.

## Properties

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">Database</a>	Used to set or return the database name.
<a href="#">Eof</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">GlobalAction</a>	Used to set or return the global transaction action.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">LimboTransactionInfo</a>	Used to return the limbo transaction information.
<a href="#">LimboTransactionInfoCount</a>	Used to set or return the number of elements in the TIBCLimboTransactionInfo array.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">Options</a>	Used to invoke database validation.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">RecoverTwoPhaseGlobal</a>	Used to restore two-phase transactions.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.

## Methods

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.

<a href="#">FetchLimboTransactionInfo</a>	Gets limbo transaction information.
<a href="#">FixLimboTransactionErrors</a>	Fixes limbo transaction errors.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Returns the next line of data.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.
<a href="#">SweepDatabase</a>	Performs a database sweep.

## Events

Name	Description
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.20.2 Properties

Properties of the **TIBCVValidationService** class.  
For a complete list of the **TIBCVValidationService** class members, see the [TIBCVValidationService Members](#) topic.

## Public

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Eof</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBCCControlAndQueryService</a> )	Returns the next line of data.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">LimboTransactionInfo</a>	Used to return the limbo transaction information.
<a href="#">LimboTransactionInfoCount</a>	Used to set or return the number of elements in the TIBCLimboTransactionInfo array.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.

## Published

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">Database</a>	Used to set or return the database name.
<a href="#">GlobalAction</a>	Used to set or return the global transaction action.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Options</a>	Used to invoke database validation.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">RecoverTwoPhaseGlobal</a>	Used to restore two-phase transactions.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.

## See Also

- [TIBCVValidationService Class](#)
- [TIBCVValidationService Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.20.2.1 Database Property

Used to set or return the database name.

## Class

[TIBCVValidationService](#)

## Syntax

```
property Database: string;
```

## Remarks

Use the Database property to set or return the database name to set properties on.

© 1997-2013 Devart. All Rights Reserved.



## 17.14.1.20.2.2 GlobalAction Property

Used to set or return the global transaction action.

**Class**

[TIBCValidationService](#)

**Syntax**

```
property GlobalAction: TIBCTransactionGlobalAction default
    tgNoGlobalAction;
```

**Remarks**

Use the GlobalAction property to set or return the global transaction action. This value indicates what action to take with the transactions that follow.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.20.2.3 LimboTransactionInfo Property(Indexer)

Used to return the limbo transaction information.

**Class**

[TIBCValidationService](#)

**Syntax**

```
property LimboTransactionInfo[Index: Integer]:
    TIBCLimboTransactionInfo;
```

**Parameters**

*Index*

**Remarks**

Use the LimboTransactionInfo property to return the limbo transaction information.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.20.2.4 LimboTransactionInfoCount Property

Used to set or return the number of elements in the TIBCLimboTransactionInfo array.

**Class**

[TIBCValidationService](#)

**Syntax**

```
property LimboTransactionInfoCount: Integer;
```

**Remarks**

Use the LimboTransactionInfoCount property to set or return the number of elements in the TIBCLimboTransactionInfo array.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.20.2.5 Options Property

Used to invoke database validation.

### Class

[TIBCValidationService](#)

### Syntax

```
property Options: TIBCValidateOptions default [];
```

### Remarks

Use the Options property of the TIBCValidationService component to invoke database validation.

Set any of the following options of type TValidateOption to True to perform the appropriate validation:

**Note:** Not all combinations of validation options work together. For example, you can not simultaneously mend and validate the database at the same time.

Conversely, some options are intended to be used with other options, such as ValidateDB, or ValidateFull with ValidateDB.

### Example

To set these options in code, use the Options property:

```
Options := [CheckDB, IgnoreChecksum, KillShadows];
```

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.20.2.6 RecoverTwoPhaseGlobal Property

Used to restore two-phase transactions.

### Class

[TIBCValidationService](#)

### Syntax

```
property RecoverTwoPhaseGlobal: Boolean;
```

### Remarks

Use the RecoverTwoPhaseGlobal property to restore two-phase transactions.

RecoverTwoPhaseGlobal performs automated two-phase recovery, either for a limbo transaction specified by ID or for all limbo transactions.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.20.3 Methods

Methods of the **TIBCValidationService** class.

For a complete list of the **TIBCValidationService** class members, see the [TIBCValidationService Members](#) topic.

**Public**

Name	Description
<a href="#">Attach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Attaches to the database.
<a href="#">Detach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Detaches from the database.
<a href="#">Eof</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to determine whether the end of the file has been reached.
<a href="#">FetchLimboTransactionInfo</a>	Gets limbo transaction information.
<a href="#">FixLimboTransactionErrors</a>	Fixes limbo transaction errors.
<a href="#">GetNextChunk</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next chunk of data.
<a href="#">GetNextLine</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Returns the next line of data.
<a href="#">Handle</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return the database handle.
<a href="#">ServiceParamBySPB</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to return and sets SPB parameters.
<a href="#">ServiceStart</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Starts the service.
<a href="#">SweepDatabase</a>	Performs a database sweep.

**Published**

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the service to active or inactive (default).
<a href="#">BufferSize</a> (inherited from <a href="#">TIBControlAndQueryService</a> )	Used to set or return the buffer size.
<a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to display a login prompt before attaching to a database.
<a href="#">OnAttach</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Occurs when the database is attached.
<a href="#">Params</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set or return database parameters.
<a href="#">Protocol</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to select the network protocol.
<a href="#">Server</a> (inherited from <a href="#">TCustomIBCSERVICE</a> )	Used to set the name of the server on which the services are to be run.

**See Also**

- [TIBCValidationService Class](#)
  - [TIBCValidationService Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.20.3.1 FetchLimboTransactionInfo Method

Gets limbo transaction information.

##### **Class**

[TIBCValidationService](#)

##### **Syntax**

```
procedure FetchLimboTransactionInfo;
```

##### **Remarks**

Call the FetchLimboTransactionInfo method to get limbo transaction information from the [LimboTransactionInfo](#) property.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.20.3.2 FixLimboTransactionErrors Method

Fixes limbo transaction errors.

##### **Class**

[TIBCValidationService](#)

##### **Syntax**

```
procedure FixLimboTransactionErrors;
```

##### **Remarks**

Call the FixLimboTransactionErrors method to repair limbo transaction errors.

**Note:** You should install InterBase 6 to use this feature.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.20.3.3 SweepDatabase Method

Performs a database sweep.

##### **Class**

[TIBCValidationService](#)

##### **Syntax**

```
procedure SweepDatabase;
```

##### **Remarks**

Call the SweepDatabase method to perform a database sweep.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.14.1.2 TIBCVersionInfo Class

Represents the version information about an InterBase server.  
For a list of all members of this type, see [TIBCVersionInfo](#) members.

#### Unit

[IBCAAdmin](#)

#### Syntax

```
TIBCVersionInfo = class (System.TObject);
```

#### Remarks

TIBCVersionInfo represents the version information about an InterBase server.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.21.1 Members

[TIBCVersionInfo](#) class overview.

#### Properties

Name	Description
<a href="#">ServerImplementation</a>	Used to retrieve the implementation string.
<a href="#">ServerVersion</a>	Used to retrieve the version of the InterBase server.
<a href="#">ServiceVersion</a>	Used to retrieve the version number for the service that fetches the version information.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.1.21.2 Properties

Properties of the **TIBCVersionInfo** class.  
For a complete list of the **TIBCVersionInfo** class members, see the [TIBCVersionInfo Members](#) topic.

#### Public

Name	Description
<a href="#">ServerImplementation</a>	Used to retrieve the implementation string.
<a href="#">ServerVersion</a>	Used to retrieve the version of the InterBase server.
<a href="#">ServiceVersion</a>	Used to retrieve the version number for the service that fetches the version information.

**See Also**

- [TIBCVersionInfo Class](#)
  - [TIBCVersionInfo Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.21.2.1 ServerImplementation Property

Used to retrieve the implementation string.

**Class**

[TIBCVersionInfo](#)

**Syntax**

```
property ServerImplementation: string;
```

**Remarks**

Use the ServerImplementation property to retrieve the implementation string.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.21.2.2 ServerVersion Property

Used to retrieve the version of the InterBase server.

**Class**

[TIBCVersionInfo](#)

**Syntax**

```
property ServerVersion: string;
```

**Remarks**

Use the ServerVersion property to retrieve the version of the InterBase server.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.1.21.2.3 ServiceVersion Property

Used to retrieve the version number for the service that fetches the version information.

**Class**

[TIBCVersionInfo](#)

**Syntax**

```
property ServiceVersion: integer;
```

**Remarks**

Use the ServiceVersion property to retrieve the version number for the service that fetches the version information.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.14.2 Types

Types in the **IBCAdmin** unit.

### Types

Name	Description
<a href="#">TIBCBackupOptions</a>	Represents the set of <a href="#">TIBCBackupOption</a> .
<a href="#">TIBCNBackupOptions</a>	Represents the set of <a href="#">TIBCNBackupOption</a> .
<a href="#">TIBCRestoreOptions</a>	Represents the set of <a href="#">TIBCRestoreOption</a> .
<a href="#">TIBCStatOptions</a>	Represents the set of <a href="#">TIBCStatOption</a> .
<a href="#">TIBCValidateOptions</a>	Represents the set of <a href="#">TIBCValidateOption</a> .

© 1997-2013 Devart. All Rights Reserved.

### 17.14.2.1 TIBCBackupOptions Set

Represents the set of [TIBCBackupOption](#).

#### Unit

[IBCAdmin](#)

#### Syntax

```
TIBCBackupOptions = set of TIBCBackupOption;
```

© 1997-2013 Devart. All Rights Reserved.

### 17.14.2.2 TIBCNBackupOptions Set

Represents the set of [TIBCNBackupOption](#).

#### Unit

[IBCAdmin](#)

#### Syntax

```
TIBCNBackupOptions = set of TIBCNBackupOption;
```

© 1997-2013 Devart. All Rights Reserved.

### 17.14.2.3 TIBCRestoreOptions Set

Represents the set of [TIBCRestoreOption](#).

#### Unit

[IBCAdmin](#)

**Syntax**

```
TIBCRestoreOptions = set of TIBCRestoreOption;
```

© 1997-2013 Devart. All Rights Reserved.

**17.14.2.4 TIBCStatOptions Set**

Represents the set of [TIBCStatOption](#).

**Unit**

[IBCAdmin](#)

**Syntax**

```
TIBCStatOptions = set of TIBCStatOption;
```

© 1997-2013 Devart. All Rights Reserved.

**17.14.2.5 TIBCValidateOptions Set**

Represents the set of [TIBCValidateOption](#).

**Unit**

[IBCAdmin](#)

**Syntax**

```
TIBCValidateOptions = set of TIBCValidateOption;
```

© 1997-2013 Devart. All Rights Reserved.

**17.14.3 Enumerations**

Enumerations in the **IBCAdmin** unit.

**Enumerations**

Name	Description
<a href="#">TIBCBackupOption</a>	Allows you to build backup options into your application.
<a href="#">TIBCLicensingAction</a>	Allows to add or remove an InterBase software activation certificate.
<a href="#">TIBCNBackupOption</a>	Options used in <a href="#">TIBCBackupRestoreService</a> for nBackup.
<a href="#">TIBCRestoreOption</a>	Specifies the data restore parameters.
<a href="#">TIBCSecurityAction</a>	Specify the type of the operation for InterBase Security Service to perform.



<a href="#">TIBCStatOption</a>	Allows to specify the way the database statistics would be requested.
<a href="#">TIBCTransactionAdvise</a>	Allows to specify the suggested action for ending the transaction.
<a href="#">TIBCTransactionGlobalAction</a>	Determines which action to take concerning limbo transactions.
<a href="#">TIBCTransactionState</a>	Allows to specify the current state of the limbo transaction.
<a href="#">TIBCValidateOption</a>	Sets the options of validation.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.3.1 TIBCBackupOption Enumeration

Allows you to build backup options into your application.

##### Unit

[IBCAdmin](#)

##### Syntax

```
TIBCBackupOption = (boIgnoreChecksums, boIgnoreLimbo,
    boMetadataOnly, boNoGarbageCollection, boOldMetadataDesc,
    boNonTransportable, boConvertExtTables);
```

##### Values

Value	Meaning
<b>boConvertExtTables</b>	Convert external table data to internal tables.
<b>boIgnoreChecksums</b>	Ignore checksums during backup.
<b>boIgnoreLimbo</b>	Ignored limbo transactions during backup.
<b>boMetadataOnly</b>	Output backup file for metadata only with empty tables.
<b>boNoGarbageCollection</b>	Suppress normal garbage collection during backup; improves performance on some databases.
<b>boNonTransportable</b>	Output backup file with non-XDR data format; improves space and performance by a negligible amount.
<b>boOldMetadataDesc</b>	Output metadata in pre-4.0 format.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.3.2 TIBCLicensingAction Enumeration

Allows to add or remove an InterBase software activation certificate.

##### Unit

[IBCAdmin](#)

##### Syntax

```
TIBCLicensingAction = (laAdd, laRemove);
```

##### Values

Value	Meaning
<b>laAdd</b>	Adds an InterBase software activation certificate.
<b>laRemove</b>	Removes an InterBase software activation certificate.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.3.3 TIBCNBackupOption Enumeration

Options used in [TIBCBBackupRestoreService](#) for nBackup.

##### Unit

[IBCAdmin](#)

##### Syntax

```
TIBCNBackupOption = (nboNoTriggers);
```

##### Values

Value	Meaning
<b>nboNoTriggers</b>	Disables triggers while performing backup or restore.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.3.4 TIBCRestoreOption Enumeration

Specifies the data restore parameters.

##### Unit


[IBCAdmin](#)

##### Syntax

```
TIBCRestoreOption = (roDeactivateIndexes, roNoShadow,  
roNoValidityCheck, roOneRelationAtATime, roReplace,  
roCreateNewDB, roUseAllSpace, roValidationCheck);
```

##### Values

Value	Meaning
<b>roCreateNewDB</b>	Restore but do not overwrite an existing database (restores the database from the backup copy into a new file).

<b>roDeactivateIndexes</b>	Do not build indexes during restore. InterBase rebuilds indexes while restoring a database in the normal mode of operation. If there are duplicates of the records that are unique in the database, the restore process will be terminated. Duplicate records can be added to the database if the unique index is temporary turned off.
<b>roNoShadow</b>	Do not recreate shadow files during restore. In the normal mode of operation InterBase recreates shadow files during restore. By setting this value such option will be blocked.
<b>roNoValidityCheck</b>	Do not enforce validity conditions (for example, NOT NULL) during restore. This value should be used when restoring a database that includes records in a wrong format. If you set this value, the correspondence of data to the existing limitations would not be checked.
<b>roOneRelationAtATime</b>	Commit after completing a restore of each table.
<b>roReplace</b>	Replace database if one exists. 
<b>roUseAllSpace</b>	Do not reserve 20% of each datapage for future record versions; useful for read-only databases.
<b>roValidationCheck</b>	Perform validation check before restoring a database.

© 1997-2013 Devart. All Rights Reserved.

17.14.3.5 TIBCSecurityAction Enumeration

Specify the type of the operation for InterBase Security Service to perform.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCSecurityAction = (saAddUser, saDeleteUser, saModifyUser, saDisplayUser);
```

Values

Value	Meaning
<b>saAddUser</b>	New user account will be added.
<b>saDeleteUser</b>	Existing user account will be deleted.
<b>saDisplayUser</b>	Available information on the user account will be displayed.
<b>saModifyUser</b>	The user account will be modified.

© 1997-2013 Devart. All Rights Reserved.

**17.14.3.6 TIBStatOption Enumeration**

Allows to specify the way the database statistics would be requested.

**Unit**

[IBCAdmin](#)

**Syntax**

```
TIBStatOption = (soDataPages, soDbLog, soHeaderPages,
  soIndexPages, soSystemRelations, soRecordVersions,
  soStatTables);
```

**Values**

Value	Meaning
<b>soDataPages</b>	Request statistics for data tables in the database.
<b>soDbLog</b>	Stop reporting statistics after reporting information on the log pages.
<b>soHeaderPages</b>	Stop reporting statistics after reporting the information on the header page.
<b>soIndexPages</b>	Request statistics for the user indexes in the database.
<b>soRecordVersions</b>	Request statistics for the record versions.
<b>soStatTables</b>	Request statistics for tables specified in the <a href="#">TableNames</a> property.
<b>soSystemRelations</b>	Request statistics for system tables and indexes in addition to user tables and indexes.

---

© 1997-2013 Devart. All Rights Reserved.

**17.14.3.7 TIBTransactionAdvise Enumeration**

Allows to specify the suggested action for ending the transaction.

**Unit**

[IBCAdmin](#)

**Syntax**

```
TIBTransactionAdvise = (taCommitAdvise, taRollbackAdvise,
  taUnknownAdvise);
```

**Values**

Value	Meaning
<b>taCommitAdvise</b>	Commit the transaction.
<b>taRollbackAdvise</b>	Rollback the transaction.
<b>taUnknownAdvise</b>	No suggested action is specified for ending the transaction.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.3.8 TIBCTransactionGlobalAction Enumeration

Determines which action to take concerning limbo transactions.

##### Unit

[IBCAdmin](#)

##### Syntax

```
TIBCTransactionGlobalAction = (tgNoGlobalAction, tgCommitGlobal,
    tgRollbackGlobal);
```

##### Values

Value	Meaning
<b>tgCommitGlobal</b>	Commits the limbo transaction specified by ID or commits all limbo transactions.
<b>tgNoGlobalAction</b>	Takes no action.
<b>tgRollbackGlobal</b>	Rolls back the limbo transaction specified by ID or rolls back all limbo transactions.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.3.9 TIBCTransactionState Enumeration

Allows to specify the current state of the limbo transaction.

##### Unit

[IBCAdmin](#)

##### Syntax

```
TIBCTransactionState = (tsLimboState, tsCommitState,
    tsRollbackState, tsUnknownState);
```

##### Values

Value	Meaning
<b>tsCommitState</b>	The transaction is in commit state.
<b>tsLimboState</b>	The limbo transaction is in limbo state.
<b>tsRollbackState</b>	The limbo transaction is in rollback state.
<b>tsUnknownState</b>	The state if the limbo transaction is unknown.

© 1997-2013 Devart. All Rights Reserved.

#### 17.14.3.1 TIBCValidateOption Enumeration

Sets the options of validation.

##### Unit

[IBCAdmin](#)

##### Syntax

```
TIBCValidateOption = (voCheckDB, voIgnoreChecksum, voKillShadows,
```

```
voMendDB, voValidateDB, voValidateFull);
```

### Values

Value	Meaning
<b>voCheckDB</b>	Request a read-only validation of the database without correcting any problems.
<b>voIgnoreChecksum</b>	Ignore all checksum errors when validating or sweeping.
<b>voKillShadows</b>	Remove references to unavailable shadow files.
<b>voMendDB</b>	Mark corrupted records as unavailable so that subsequent operations skip them.
<b>voValidateDB</b>	Locate and release pages that are allocated but unassigned to any data structures.
<b>voValidateFull</b>	Check record and page structures, releasing unassigned record fragments; use with <i>ValidateDB</i> .

© 1997-2013 Devart. All Rights Reserved.

## 17.15 IBCAlerter

This unit contains implementation of the TIBCAlerter component.

### Classes

Name	Description
<a href="#">TIBCAlerter</a>	A component for transferring messages between connections.

### Types

Name	Description
<a href="#">TIBCAAlertEvent</a>	This type is used for the <a href="#">TIBCAlerter.OnEvent</a> event.

© 1997-2013 Devart. All Rights Reserved.

### 17.15.1 Classes

Classes in the **IBCAlerter** unit.

### Classes

Name	Description
<a href="#">TIBCAlerter</a>	A component for transferring messages between connections.

© 1997-2013 Devart. All Rights Reserved.

### 17.15.1.1 TIBCAlerter Class

A component for transferring messages between connections.  
For a list of all members of this type, see [TIBCAlerter](#) members.

#### Unit

[IBCAlerter](#)

#### Syntax

```
TIBCAlerter = class(TDAAlerter);
```

#### Remarks

The TIBCAlerter component allows you to register interest in and asynchronously handle events posted by an InterBase server. Use TIBCAlerter to handle events for responding to actions and database changes made by other applications. To get events application must register required events. To do it set the [TIBCAlerter.Events](#) property to required events and call M:Devart.IbDac.TIBCAlerter.Start() method. When one of the registered events occurs [TIBCAlerter.OnEvent](#) handler is called.

**Note:** Events are transaction-based. This means that the waiting connection does not get event until the transaction posting the event commits.

#### Inheritance Hierarchy

[TDAAlerter](#)  
**TIBCAlerter**

© 1997-2013 Devart. All Rights Reserved.

#### 17.15.1.1.1 Members

[TIBCAlerter](#) class overview.

#### Properties

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TDAAlerter</a> )	Used to determine if TDAAlerter waits for messages.
<a href="#">AutoCommit</a>	Used to automatically commit transaction after calling the SendEvent method.
<a href="#">AutoRegister</a> (inherited from <a href="#">TDAAlerter</a> )	Used to automatically register events whenever connection opens.
<a href="#">Connection</a>	Used to specify the connection for TIBCAlerter.
<a href="#">Events</a>	Used to specify the names of events to wait.
<a href="#">Transaction</a>	Used to set the transaction to be used by the SendEvent method.

## Methods

Name	Description
<a href="#">SendEvent</a>	Sends an event with Name.
<a href="#">Start</a> (inherited from <a href="#">TDAAlerter</a> )	Starts waiting process.
<a href="#">Stop</a> (inherited from <a href="#">TDAAlerter</a> )	Stops waiting process.

## Events

Name	Description
<a href="#">OnError</a> (inherited from <a href="#">TDAAlerter</a> )	Occurs if an exception occurs in waiting process
<a href="#">OnEvent</a>	Occurs when waiting process receives an event from the InterBase server.

© 1997-2013 Devart. All Rights Reserved.

### 17.15.1.1.2 Properties

Properties of the **TIBCAlerter** class.

For a complete list of the **TIBCAlerter** class members, see the [TIBCAlerter Members](#) topic.

## Public

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TDAAlerter</a> )	Used to determine if TDAAlerter waits for messages.
<a href="#">AutoRegister</a> (inherited from <a href="#">TDAAlerter</a> )	Used to automatically register events whenever connection opens.
<a href="#">OnError</a> (inherited from <a href="#">TDAAlerter</a> )	Occurs if an exception occurs in waiting process
<a href="#">SendEvent</a> (inherited from <a href="#">TDAAlerter</a> )	Sends an event with Name and content Message.
<a href="#">Start</a> (inherited from <a href="#">TDAAlerter</a> )	Starts waiting process.
<a href="#">Stop</a> (inherited from <a href="#">TDAAlerter</a> )	Stops waiting process.

## Published

Name	Description
<a href="#">AutoCommit</a>	Used to automatically commit transaction after calling the SendEvent method.
<a href="#">Connection</a>	Used to specify the connection for TIBCAlerter.
<a href="#">Events</a>	Used to specify the names of events to wait.



## [Transaction](#)

Used to set the transaction to be used by the SendEvent method.

### See Also

- [TIBCAlerter Class](#)
- [TIBCAlerter Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.15.1.1.2.1 AutoCommit Property

Used to automatically commit transaction after calling the SendEvent method.

### Class

[TIBCAlerter](#)

### Syntax

```
property AutoCommit: boolean;
```

### Remarks

Use the AutoCommit property to automatically commit transaction after calling the SendEvent method. Events are transaction-based. This means that the waiting connection does not get event until the transaction posting the event commits.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.15.1.1.2.2 Connection Property

Used to specify the connection for TIBCAlerter.

### Class

[TIBCAlerter](#)

### Syntax

```
property Connection: TIBCCConnection;
```

### Remarks

Use the Connection property to specify the connection for TIBCAlerter.

### See Also

- [TIBCCConnection](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.15.1.1.2.3 Events Property

Used to specify the names of events to wait.

### Class

## [TIBCAlerter](#)

### Syntax

**property** Events: `_TStrings;`

### Remarks

Use the Events property to set the names of events to wait.

© 1997-2013 Devart. All Rights Reserved.

#### 17.15.1.1.2.4 Transaction Property

Used to set the transaction to be used by the SendEvent method.

### Class

## [TIBCAlerter](#)

### Syntax

**property** Transaction: [TIBCTransaction](#) **stored** IsTransactionStored;

### Remarks

Use the Transaction property to set the transaction to be used by the SendEvent method. Events are transaction-based. This means that the waiting connection does not get event until the transaction posting the event commits.

© 1997-2013 Devart. All Rights Reserved.

#### 17.15.1.1.3 Methods

Methods of the **TIBCAlerter** class.

For a complete list of the **TIBCAlerter** class members, see the [TIBCAlerter Members](#) topic.

### Public

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TDAAlerter</a> )	Used to determine if TDAAlerter waits for messages.
<a href="#">AutoRegister</a> (inherited from <a href="#">TDAAlerter</a> )	Used to automatically register events whenever connection opens.
<a href="#">Connection</a> (inherited from <a href="#">TDAAlerter</a> )	Used to specify the connection for TDAAlerter.
<a href="#">OnError</a> (inherited from <a href="#">TDAAlerter</a> )	Occurs if an exception occurs in waiting process
<a href="#">SendEvent</a>	Sends an event with Name.
<a href="#">Start</a> (inherited from <a href="#">TDAAlerter</a> )	Starts waiting process.
<a href="#">Stop</a> (inherited from <a href="#">TDAAlerter</a> )	Stops waiting process.

### See Also

- [TIBCAlerter Class](#)
- [TIBCAlerter Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.15.1.1.3.1 SendEvent Method

Sends an event with Name.

### Class

[TIBCAlerter](#)

### Syntax

```
procedure SendEvent (const Name: string) ;
```

### Parameters

*Name*

Holds the name of the event to send.

### Remarks

Call the SendEvent procedure to send an event with Name.  
This procedure is supported only for Firebird 2.0 and higher.

© 1997-2013 Devart. All Rights Reserved.

#### 17.15.1.1.4 Events

Events of the **TIBCAlerter** class.

For a complete list of the **TIBCAlerter** class members, see the [TIBCAlerter Members](#) topic.

### Public

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TDAAlerter</a> )	Used to determine if TDAAlerter waits for messages.
<a href="#">AutoRegister</a> (inherited from <a href="#">TDAAlerter</a> )	Used to automatically register events whenever connection opens.
<a href="#">Connection</a> (inherited from <a href="#">TDAAlerter</a> )	Used to specify the connection for TDAAlerter.
<a href="#">OnError</a> (inherited from <a href="#">TDAAlerter</a> )	Occurs if an exception occurs in waiting process
<a href="#">SendEvent</a> (inherited from <a href="#">TDAAlerter</a> )	Sends an event with Name and content Message.
<a href="#">Start</a> (inherited from <a href="#">TDAAlerter</a> )	Starts waiting process.
<a href="#">Stop</a> (inherited from <a href="#">TDAAlerter</a> )	Stops waiting process.

### Published

Name	Description
------	-------------

[OnEvent](#)

Occurs when waiting process receives an event from the InterBase server.

**See Also**

- [TIBCAlerter Class](#)
- [TIBCAlerter Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.15.1.1.4.1 OnEvent Event

Occurs when waiting process receives an event from the InterBase server.

**Class**

[TIBCAlerter](#)

**Syntax**

**property** OnEvent: [TIBCAAlertEvent](#);

**Remarks**

The OnEvent event occurs when waiting process receives event from the InterBase server. The EventName parameter contains the name of the event and the EventCount parameter contains the number of events that were raised in the transaction. Waiting connection does not get event until the transaction signaling the alert commits and there can be several event raises in one transaction.

© 1997-2013 Devart. All Rights Reserved.

**17.15.2 Types**

Types in the **IBCAlerter** unit.

**Types**

Name	Description
<a href="#">TIBCAAlertEvent</a>	This type is used for the <a href="#">TIBCAlerter.OnEvent</a> event.

© 1997-2013 Devart. All Rights Reserved.

**17.15.2.1 TIBCAAlertEvent Procedure Reference**

This type is used for the [TIBCAlerter.OnEvent](#) event.

**Unit**

[IBCAlerter](#)

**Syntax**

```
TIBCAAlertEvent = procedure (Sender: TObject; EventName: string;  
EventCount: Integer) of object;
```

**Parameters**

*Sender*

An object that raised the event.

*EventName*

Holds the name of the event.

*EventCount*

Holds the number of events that were raised in the transaction.

© 1997-2013 Devart. All Rights Reserved.

## 17.16 IBCArray

This unit contains the TIBCArray class for representing the value of the InterBase array data type.

### Classes

Name	Description
<a href="#">TCustomIBCArray</a>	A base class representing the value of the InterBase array data type.

© 1997-2013 Devart. All Rights Reserved.

### 17.16.1 Classes

Classes in the **IBCArray** unit.

### Classes

Name	Description
<a href="#">TCustomIBCArray</a>	A base class representing the value of the InterBase array data type.

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1 TCustomIBCArray Class

A base class representing the value of the InterBase array data type.  
For a list of all members of this type, see [TCustomIBCArray](#) members.

### Unit

[IBCArray](#)

### Syntax

```
TCustomIBCArray = class (TDBObject) ;
```

### Inheritance Hierarchy

[TSharedObject](#)

[TDBObject](#)

**TCustomIBCArray**

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.1 Members

[TCustomIBCArray](#) class overview.

### Properties

Name	Description
<a href="#">ArrayDimensions</a>	Contains the array dimensions count.
<a href="#">ArrayHighBound</a>	Used to get or set the upper boundary of the defined dimension subscript.
<a href="#">ArrayID</a>	Contains the array ID.
<a href="#">ArrayLowBound</a>	Used to get or set the lower boundary of the defined dimension subscript.
<a href="#">ArraySize</a>	Used to determine the size of the whole array data in bytes.
<a href="#">AsString</a>	Used to return array as string.
<a href="#">Cached</a>	Indicates whether to cache array data on the client side.
<a href="#">CachedDimensions</a>	Contains cached array dimensions count.
<a href="#">CachedHighBound</a>	Used to get the upper boundary of defined dimension subscript of cached array elements.
<a href="#">CachedLowBound</a>	Used to get the lower boundary of the defined dimension subscript of cached array elements.
<a href="#">CachedSize</a>	Used to get the cached array data size in bytes.
<a href="#">ColumnName</a>	Used to get or set the name of the table column that has array type.
<a href="#">DbHandle</a>	Contains the handle of a database where the array is stored.
<a href="#">IsNull</a>	Used to define whether the array field in the database is null.
<a href="#">Items</a>	Used to get or set array items.

<a href="#">ItemScale</a>	Used to get or set the scale for array items for the NUMERIC and DECIMAL datatypes.
<a href="#">ItemSize</a>	Contains the size of an array item.
<a href="#">Modified</a>	Used to indicate if the modifications done in cache were saved to the database.
<a href="#">RefCount</a> (inherited from <a href="#">TSharedObject</a> )	Used to return the count of reference to a TSharedObject object.
<a href="#">TableName</a>	Used to set the name of the table containing an array field.
<a href="#">TrHandle</a>	Contains the handle of the transaction in which the array is read or written.

## Methods

Name	Description
<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">Assign</a>	Copies the Source object content to the current one.
<a href="#">ClearArray</a>	Clears all array values on the server if Cached is set to False.
<a href="#">CreateTemporaryArray</a>	Creates a temporary array on the InterBase server.
<a href="#">GetArrayInfo</a>	Used to get array descriptor.
<a href="#">GetItemAsDateTime</a>	Reads the value of an array item into an object or variable of the TDateTime type.
<a href="#">GetItemAsFloat</a>	Reads the value of an array item into a floating-point number.
<a href="#">GetItemAsInteger</a>	Reads the value of an array item into an integer.
<a href="#">GetItemAsSmallInt</a>	Reads the value of an array item into a short integer.
<a href="#">GetItemAsString</a>	Reads the value of an array item into a string.
<a href="#">GetItemAsWideString</a>	Reads the value of an array item into a WideString.

<a href="#">GetItemsSlice</a>	Returns the array slice items' values.
<a href="#">GetItemValue</a>	Returns the array item value.
<a href="#">ReadArray</a>	Reads an array from the database to memory.
<a href="#">ReadArrayItem</a>	Reads the array item specified by indices from the database to memory.
<a href="#">ReadArraySlice</a>	Reads array slice from the database to memory.
<a href="#">Release</a> (inherited from <a href="#">TSharedObject</a> )	Decrements the reference count.
<a href="#">SetItemAsDateTime</a>	Assigns the TDateTime value to the contents of an array item.
<a href="#">SetItemAsFloat</a>	Assigns floating-point value to the contents of an array item.
<a href="#">SetItemAsInteger</a>	Assigns integer value to the contents of an array item.
<a href="#">SetItemAsSmallInt</a>	Assigns short integer value to the contents of an array item.
<a href="#">SetItemAsString</a>	Assigns string value to the contents of an array item.
<a href="#">SetItemAsWideString</a>	Assigns WideString value to the contents of an array item.
<a href="#">SetItemsSlice</a>	Sets the array slice values.
<a href="#">SetItemValue</a>	Sets the array item value.
<a href="#">WriteArray</a>	Writes all cached array values to the database.
<a href="#">WriteArraySlice</a>	Writes cached array slice.

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.2 Properties

Properties of the **TCustomIBCArrary** class.

For a complete list of the **TCustomIBCArrary** class members, see the [TCustomIBCArrary Members](#) topic.

#### Public

Name	Description
<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.



<a href="#"><u>ArrayDimensions</u></a>	Contains the array dimensions count.
<a href="#"><u>ArrayHighBound</u></a>	Used to get or set the upper boundary of the defined dimension subscript.
<a href="#"><u>ArrayID</u></a>	Contains the array ID.
<a href="#"><u>ArrayLowBound</u></a>	Used to get or set the lower boundary of the defined dimension subscript.
<a href="#"><u>ArraySize</u></a>	Used to determine the size of the whole array data in bytes.
<a href="#"><u>AsString</u></a>	Used to return array as string.
<a href="#"><u>Cached</u></a>	Indicates whether to cache array data on the client side.
<a href="#"><u>CachedDimensions</u></a>	Contains cached array dimensions count.
<a href="#"><u>CachedHighBound</u></a>	Used to get the upper boundary of defined dimension subscript of cached array elements.
<a href="#"><u>CachedLowBound</u></a>	Used to get the lower boundary of the defined dimension subscript of cached array elements.
<a href="#"><u>CachedSize</u></a>	Used to get the cached array data size in bytes.
<a href="#"><u>ColumnName</u></a>	Used to get or set the name of the table column that has array type.
<a href="#"><u>DbHandle</u></a>	Contains the handle of a database where the array is stored.
<a href="#"><u>IsNull</u></a>	Used to define whether the array field in the database is null.
<a href="#"><u>Items</u></a>	Used to get or set array items.
<a href="#"><u>ItemScale</u></a>	Used to get or set the scale for array items for the NUMERIC and DECIMAL datatypes.
<a href="#"><u>ItemSize</u></a>	Contains the size of an array item.
<a href="#"><u>Modified</u></a>	Used to indicate if the modifications done in cache were saved to the database.

[RefCount](#) (inherited from [TSharedObject](#))

Used to return the count of reference to a TSharedObject object.

[Release](#) (inherited from [TSharedObject](#))

Decrements the reference count.

[TableName](#)

Used to set the name of the table containing an array field.

[TrHandle](#)

Contains the handle of the transaction in which the array is read or written.

### See Also

- [TCustomIBCArrary Class](#)
  - [TCustomIBCArrary Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.2.1 ArrayDimensions Property

Contains the array dimensions count.

### Class

[TCustomIBCArrary](#)

### Syntax

```
property ArrayDimensions: integer;
```

### Remarks

The ArrayDimensions property is used to hold the array dimensions count. InterBase supports multi-dimensional arrays with 1 to 16 dimensions.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.2.2 ArrayHighBound Property(Indexer)

Used to get or set the upper boundary of the defined dimension subscript.

### Class

[TCustomIBCArrary](#)

### Syntax

```
property ArrayHighBound[Dimension: integer]: integer;
```

#### Parameters

*Dimension*

Holds the dimension subscript.

### Remarks

Use the ArrayHighBound property to get or set the upper boundary of the defined dimension subscript.

## See Also

- [ArrayLowBound](#)
- [ArrayDimensions](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.16.1.1.2.3 ArrayID Property

Contains the array ID.

## Class

[TCustomIBCArrary](#)

## Syntax

```
property ArrayID: TISC_QUAD;
```

## Remarks

The ArrayID property is used to hold the array ID that is actually stored in the array field of the table record. It is a unique numeric value that references the array data.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.16.1.1.2.4 ArrayLowBound Property(Indexer)

Used to get or set the lower boundary of the defined dimension subscript.

## Class

[TCustomIBCArrary](#)

## Syntax

```
property ArrayLowBound[Dimension: integer]: integer;
```

### Parameters

*Dimension*

Holds the dimension subscript.

## Remarks

Use the ArrayLowBound property to get or set the lower boundary of the defined dimension subscript.

## See Also

- [ArrayHighBound](#)
- [ArrayDimensions](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.2.5 ArraySize Property

Used to determine the size of the whole array data in bytes.

**Class**

[TCustomIBCArrary](#)

**Syntax**

```
property ArraySize: integer;
```

**Remarks**

Use the ArraySize property to find out the size of the whole array data in bytes.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.2.6 AsString Property

Used to return array as string.

**Class**

[TCustomIBCArrary](#)

**Syntax**

```
property AsString: string;
```

**Remarks**

Use the AsString property to return array as string. For example, AsString property for two-dimensional array of integer with 2 rows, 3 elements in width can have the following value:

'((1; 2; 3), (4; 5; 6))'. Array values can be set using this property.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.2.7 Cached Property

Indicates whether to cache array data on the client side.

**Class**

[TCustomIBCArrary](#)

**Syntax**

```
property Cached: boolean;
```

**Remarks**

The Cached property is used to define whether to cache array data on the client side.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.2.8 CachedDimensions Property

Contains cached array dimensions count.

**Class**

[TCustomIBCArrary](#)

**Syntax**

```
property CachedDimensions: integer;
```

**Remarks**

The CachedDimensions property is used to hold the cached array dimensions count.

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.2.9 CachedHighBound Property(Indexer)

Used to get the upper boundary of defined dimension subscript of cached array elements.

**Class**

[TCustomIBCArrary](#)

**Syntax**

```
property CachedHighBound[Dimension: integer]: integer;
```

**Parameters**

*Dimension*

Holds the dimension subscript.

**Remarks**

Use the CachedHighBound property to get the upper boundary of defined dimension subscript of cached array elements.

**See Also**

- [ArrayHighBound](#)
- [CachedLowBound](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.2.10 CachedLowBound Property(Indexer)

Used to get the lower boundary of the defined dimension subscript of cached array elements.

**Class**

[TCustomIBCArrary](#)

**Syntax**

```
property CachedLowBound[Dimension: integer]: integer;
```

**Parameters***Dimension*

Holds the dimension subscript.

**Remarks**

Use `CachedLowBound` property to get the lower boundary of the defined dimension subscript of cached array elements.

**See Also**

- [ArrayLowBound](#)
  - [CachedHighBound](#)
- 

© 1997-2013 Devart. All Rights Reserved.

17.16.1.1.2.11 `CachedSize` Property

Used to get the cached array data size in bytes.

**Class**

[TCustomIBCArrary](#)

**Syntax**

```
property CachedSize: integer;
```

**Remarks**

Use the `CachedSize` property to get the cached array data size in bytes.

---

© 1997-2013 Devart. All Rights Reserved.

17.16.1.1.2.12 `ColumnName` Property

Used to get or set the name of the table column that has array type.

**Class**

[TCustomIBCArrary](#)

**Syntax**

```
property ColumnName: string;
```

**Remarks**

Use the `ColumnName` property to get or set the name of the table column that has array type.

**See Also**

- [GetArrayInfo](#)
  - [TableName](#)
-

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.2.13 DbHandle Property

Contains the handle of a database where the array is stored.

##### Class

[TCustomIBCArrary](#)

##### Syntax

```
property DbHandle: TISC_DB_HANDLE;
```

##### Remarks

The DbHandle property is used to hold the handle of a database where the array is stored.

##### See Also

- [TIBConnection.Handle](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.2.14 IsNull Property

Used to define whether the array field in the database is null.

##### Class

[TCustomIBCArrary](#)

##### Syntax

```
property IsNull: boolean;
```

##### Remarks

Use the IsNull property to define whether the array field in the database is null.

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.2.15 Items Property

Used to get or set array items.

##### Class

[TCustomIBCArrary](#)

##### Syntax

```
property Items: variant;
```

##### Remarks

Returns varArray, containing array items. Use the Items property to get or set array items.

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.2.16 ItemScale Property

Used to get or set the scale for array items for the NUMERIC and DECIMAL datatypes.

##### **Class**

[TCustomIBCArrary](#)

##### **Syntax**

```
property ItemScale: integer;
```

##### **Remarks**

Use the ItemScale property to get or set the scale for array items for the NUMERIC and DECIMAL datatypes.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.2.17 ItemSize Property

Contains the size of an array item.

##### **Class**

[TCustomIBCArrary](#)

##### **Syntax**

```
property ItemSize: integer;
```

##### **Remarks**

The ItemSize property is used to hold the size of an array item in bytes.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.2.18 Modified Property

Used to indicate if the modifications done in cache were saved to the database.

##### **Class**

[TCustomIBCArrary](#)

##### **Syntax**

```
property Modified: boolean;
```

##### **Remarks**

The Modified property is True when the array was modified in cache and these changes were not saved to the database.

---

© 1997-2013 Devart. All Rights Reserved.



## 17.16.1.1.2.19 TableName Property

Used to set the name of the table containing an array field.

**Class**

[TCustomIBCArray](#)

**Syntax**

```
property TableName: string;
```

**Remarks**

Use the TableName property to get or set the name of the table containing an array field.

**See Also**

- [GetArrayInfo](#)
- [ColumnName](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.2.20 TrHandle Property

Contains the handle of the transaction in which the array is read or written.

**Class**

[TCustomIBCArray](#)

**Syntax**

```
property TrHandle: TISC_TR_HANDLE;
```

**Remarks**

The TrHandle property is used to hold the handle of the transaction in which the array is read or written.

**See Also**

- [TIBCTransaction.Handle](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.3 Methods

Methods of the **TCustomIBCArray** class.

For a complete list of the **TCustomIBCArray** class members, see the [TCustomIBCArray Members](#) topic.

**Public**

Name	Description
------	-------------

[AddRef](#) (inherited from [TSharedObject](#))

Increments the reference count for the number of references dependent on the TSharedObject object.

[Assign](#)

Copies the Source object content to the current one.

[ClearArray](#)

Clears all array values on the server if Cached is set to False.

[CreateTemporaryArray](#)

Creates a temporary array on the InterBase server.

[GetArrayInfo](#)

Used to get array descriptor.

[GetItemAsDateTime](#)

Reads the value of an array item into an object or variable of the TDateTime type.

[GetItemAsFloat](#)

Reads the value of an array item into a floating-point number.

[GetItemAsInteger](#)

Reads the value of an array item into an integer.

[GetItemAsSmallInt](#)

Reads the value of an array item into a short integer.

[GetItemAsString](#)

Reads the value of an array item into a string.

[GetItemAsWideString](#)

Reads the value of an array item into a WideString.

[GetItemsSlice](#)

Returns the array slice items' values.

[GetItemValue](#)

Returns the array item value.

[ReadArray](#)

Reads an array from the database to memory.

[ReadArrayItem](#)

Reads the array item specified by indices from the database to memory.

[ReadArraySlice](#)

Reads array slice from the database to memory.

[RefCount](#) (inherited from [TSharedObject](#))

Used to return the count of reference to a TSharedObject object.

[Release](#) (inherited from [TSharedObject](#))

Decrements the reference count.

[SetItemAsDateTime](#)

Assigns the TDateTime value to the contents of an array item.

[SetItemAsFloat](#)

Assigns floating-point value to the contents of an array item.

[SetItemAsInteger](#)

Assigns integer value to the contents of an array item.

[SetItemAsSmallInt](#)

Assigns short integer value to the contents of an array item.

[SetItemAsString](#)

Assigns string value to the contents of an array item.

[SetItemAsWideString](#)

Assigns WideString value to the contents of an array item.

[SetItemsSlice](#)

Sets the array slice values.

[SetItemValue](#)

Sets the array item value.

[WriteArray](#)

Writes all cached array values to the database.

[WriteArraySlice](#)

Writes cached array slice.

### See Also

- [TCustomIBCArrary Class](#)
- [TCustomIBCArrary Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.1 Assign Method

Copies the Source object content to the current one.

### Class

[TCustomIBCArrary](#)

### Syntax

```
procedure Assign(Source: TCustomIBCArrary);
```

#### Parameters

*Source*

Holds the Source object content.

### Remarks

Use the Assign method to copy the Source object content to the current one.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.2 ClearArray Method

Clears all array values on the server if Cached is set to False.

### Class

[TCustomIBCArrary](#)

### Syntax

```
procedure ClearArray;
```

**Remarks**

Call the ClearArray method to clear all array values on the server if Cached is set to False. If Cached property is set to True, this method clears all cached array values.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.3.3 CreateTemporaryArray Method

Creates a temporary array on the InterBase server.

**Class**

[TCustomIBCArray](#)

**Syntax**

```
procedure CreateTemporaryArray;
```

**Remarks**

Call the CreateTemporaryArray method to create a temporary array on the InterBase server. To use this method, the ArrayDimensions, ItemType, ItemSize, ItemScale properties must be set.

**See Also**

- [GetArrayInfo](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.3.4 GetArrayInfo Method

Used to get array descriptor.

**Class**

[TCustomIBCArray](#)

**Syntax**

```
procedure GetArrayInfo;
```

**Remarks**

Call the GetArrayInfo method to get array descriptor, which contains information about array dimensions, high and low subscript boundaries, array item datatype, size and scale. This method is useful when using TIBCArray as IN parameter. It can be also useful for creating temporary array with the same array dimensions and item type as array field in a table. For doing this set TableName and ColumnName properties for TIBCArray object, call GetArrayInfo method and then call CreateTemporaryArray method.

**See Also**

- [CreateTemporaryArray](#)
- [TableName](#)
- [ColumnName](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.5 GetItemAsDateTime Method

Reads the value of an array item into an object or variable of the TDateTime type.

### Class

[TCustomIBCArrary](#)

### Syntax

```
function GetItemAsDateTime(Indices: array of integer): TDateTime;
```

#### Parameters

##### Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

#### Return Value

the value of an array item as DateTime.

### Remarks

Call the GetItemAsDateTime method to read the value of an array item into an object or variable of the TDateTime type.

### See Also

- [GetItemValue](#)
- [TIBCArrary.ItemType](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.6 GetItemAsFloat Method

Reads the value of an array item into a floating-point number.

### Class

[TCustomIBCArrary](#)

### Syntax

```
function GetItemAsFloat(Indices: array of integer): double;
```

#### Parameters

##### Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

#### Return Value

the value of an array item as a floating-point number.

### Remarks

Call the `GetItemAsFloat` method to read the value of an array item into a floating-point number.

### See Also

- [GetItemValue](#)
  - [TIBCArry.ItemType](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.7 GetItemAsInteger Method

Reads the value of an array item into a integer.

### Class

[TCustomIBCArry](#)

### Syntax

```
function GetItemAsInteger (Indices: array of integer): integer;
```

#### Parameters

##### *Indices*

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

#### Return Value

the value of an array item as integer.

### Remarks

Call the `GetItemAsInteger` method to read the value of an array item into a integer.

### See Also

- [GetItemValue](#)
  - [TIBCArry.ItemType](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.8 GetItemAsSmallInt Method

Reads the value of an array item into a short integer.

### Class

[TCustomIBCArry](#)

### Syntax

```
function GetItemAsSmallInt (Indices: array of integer): SmallInt;
```

### Parameters

#### Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

### Return Value

the value of an array item as short integer.

### Remarks

Call the GetItemAsSmallInt method to read the value of an array item into a short integer.

### See Also

- [GetItemValue](#)
- [TIBCArray.ItemType](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.9 GetItemAsString Method

Reads the value of an array item into a string.

### Class

[TCustomIBCArray](#)

### Syntax

```
function GetItemAsString (Indices: array of integer): string;
```

### Parameters

#### Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

### Return Value

the value of an array item as string.

### Remarks

Call the GetItemAsString method to read the value of an array item into a string.

### See Also

- [GetItemValue](#)
- [TIBCArray.ItemType](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.3.10 GetItemAsWideString Method

Reads the value of an array item into a WideString.

**Class**

[TCustomIBArray](#)

**Syntax**

```
function GetItemAsWideString(Indices: array of integer): string;
```

**Parameters***Indices*

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

**Return Value**

the value of an array item as WideString.

**Remarks**

Call the GetItemAsWideString method to read the value of an array item into a WideString.

**See Also**

- [GetItemValue](#)
  - [TIBArray.ItemType](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.3.11 GetItemsSlice Method

Returns the array slice items' values.

**Class**

[TCustomIBArray](#)

**Syntax**

```
function GetItemsSlice(Bounds: array of integer): variant;
```

**Parameters***Bounds*

Holds the upper and lower boundaries of the slice.

**Return Value**

the array slice items' values.

**Remarks**

Call the GetItemsSlice method to get the array slice items' values.

---

© 1997-2013 Devart. All Rights Reserved.



## 17.16.1.1.3.12 GetItemValue Method

Returns the array item value.

**Class**

[TCustomIBCArrary](#)

**Syntax**

```
function GetItemValue(Indices: array of integer): variant;
```

**Parameters***Indices*

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

**Return Value**

the value of an array item.

**Remarks**

Call the GetItemValue method to get the array item value.

**See Also**

- [GetItemAsDateTime](#)
- [GetItemAsFloat](#)
- [GetItemAsInteger](#)
- [GetItemAsSmallInt](#)
- [GetItemAsString](#)
- [GetItemAsWideString](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.3.13 ReadArray Method

Reads an array from the database to memory.

**Class**

[TCustomIBCArrary](#)

**Syntax**

```
procedure ReadArray;
```

**Remarks**

Call the ReadArray method to read an array from database to memory.

**See Also**

- [WriteArray](#)
- [ReadArraySlice](#)

- [ReadArrayItem](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.14 ReadArrayItem Method

Reads the array item specified by indices from the database to memory.

#### Class

[TCustomIBCArrary](#)

#### Syntax

```
procedure ReadArrayItem(Indices: array of integer);
```

#### Parameters

##### *Indices*

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

#### Remarks

Call the ReadArrayItem method to read array item specified by indices from the database to the memory.

#### See Also

- [ReadArray](#)
  - [ReadArraySlice](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.15 ReadArraySlice Method

Reads array slice from the database to memory.

#### Class

[TCustomIBCArrary](#)

#### Syntax

```
procedure ReadArraySlice(Bounds: array of integer);
```

#### Parameters

##### *Bounds*

Holds the upper and lower boundaries of the array slice.

#### Remarks

Call the ReadArraySlice method to read array slice from database to memory.

#### See Also

- [WriteArraySlice](#)

- [ReadArray](#)
- [ReadArrayItem](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.16 SetItemAsDateTime Method

Assigns the TDateTime value to the contents of an array item.

### Class

[TCustomIBArray](#)

### Syntax

```
procedure SetItemAsDateTime(Indices: array of integer; Value: TDateTime);
```

### Parameters

#### Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

#### Value

Holds the array item value as TDateTime.

### Remarks

Call the SetItemAsDateTime method to assign the TDateTime value to the contents of an array item.

### See Also

- [SetItemValue](#)
- [TIBArray.ItemType](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.17 SetItemAsFloat Method

Assigns floating-point value to the contents of an array item.

### Class

[TCustomIBArray](#)

### Syntax

```
procedure SetItemAsFloat(Indices: array of integer; Value: double);
```

### Parameters

#### Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

#### Value

Holds the array item value as floating-point.

**Remarks**

Call the SetItemAsFloat method to assign floating-point value to the contents of an array item.

**See Also**

- [SetItemValue](#)
  - [TIBCArray.ItemType](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.3.18 SetItemAsInteger Method

Assigns integer value to the contents of an array item.

**Class**

[TCustomIBCArray](#)

**Syntax**

```
procedure SetItemAsInteger(Indices: array of integer; Value:  
integer);
```

**Parameters***Indices*

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

*Value*

Holds the array item value as integer.

**Remarks**

Call the SetItemAsInteger method to assign integer value to the contents of an array item.

**See Also**

- [SetItemValue](#)
  - [TIBCArray.ItemType](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.3.19 SetItemAsSmallInt Method

Assigns short integer value to the contents of an array item.

**Class**

[TCustomIBCArray](#)

**Syntax**

```
procedure SetItemAsSmallInt(Indices: array of integer; Value: SmallInt);
```

#### Parameters

##### Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

##### Value

Holds the array item value as short integer.

#### Remarks

Call the SetItemAsSmallInt method to assign short integer value to the contents of an array item.

#### See Also

- [SetItemValue](#)
- [TIBCArray.ItemType](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.20 SetItemAsString Method

Assigns string value to the contents of an array item.

#### Class

[TCustomIBCArray](#)

#### Syntax

```
procedure SetItemAsString(Indices: array of integer; Value: string);
```

#### Parameters

##### Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

##### Value

Holds the array item value as string.

#### Remarks

Call the SetItemAsString method to assign string value to the contents of an array item.

#### See Also

- [SetItemValue](#)
- [TIBCArray.ItemType](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.3.21 SetItemAsWideString Method

Assigns WideString value to the contents of an array item.

**Class**

[TCustomIBArray](#)

**Syntax**

```
procedure SetItemAsWideString(Indices: array of integer; Value:  
string);
```

**Parameters***Indices*

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

*Value*

Holds the array item value as WideString.

**Remarks**

Call the SetItemAsWideString method to assign WideString value to the contents of an array item.

**See Also**

- [SetItemValue](#)
  - [TIBArray.ItemType](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.3.22 SetItemsSlice Method

Sets the array slice values.

**Class**

[TCustomIBArray](#)

**Syntax**

```
procedure SetItemsSlice(const Values: variant);
```

**Parameters***Values*

Holds the array slice values.

**Remarks**

Call the SetItemsSlice method to set array slice values.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.3.23 SetItemValue Method

Sets the array item value.

**Class**

[TCustomIBCArray](#)

**Syntax**

```
procedure SetItemValue(Indices: array of integer; Value: variant);
```

**Parameters***Indices*

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

*Value*

Holds the array item value.

**Remarks**

Call the SetItemValue method to set the array item value.

**See Also**

- [SetItemAsDateTime](#)
- [SetItemAsFloat](#)
- [SetItemAsInteger](#)
- [SetItemAsSmallInt](#)
- [SetItemAsString](#)
- [SetItemAsWideString](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.16.1.1.3.24 WriteArray Method

Writes all cached array values to the database.

**Class**

[TCustomIBCArray](#)

**Syntax**

```
procedure WriteArray;
```

**Remarks**

Call the WriteArray method to write all cached array values to the database. This method does nothing if the Cached property is set to False.

**See Also**

- [WriteArraySlice](#)
- [ReadArray](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.16.1.1.3.25 WriteArraySlice Method

Writes cached array slice.

### Class

[TCustomIBArray](#)

### Syntax

```
procedure WriteArraySlice(Bounds: array of integer);
```

### Parameters

*Bounds*

Holds the upper and lower boundaries of the array slice.

### Remarks

Call the WriteArraySlice method to write cached array slice. This method does nothing if Cached property is set to False.

### See Also

- [WriteArray](#)
  - [ReadArraySlice](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.17 IBCCall

Defines InterBase Call Interface routines.

### Routines

Name	Description
<a href="#">DetectGDS</a>	Searches InterBase client library and sets GDSDLL, GDSVersion variables.
<a href="#">FreeGDS</a>	Removes InterBase client library from memory.
GetIBError	Returns error messages for given error code and status vector. ErrorMessage represents the error message InterBase provides for the error. SQLErrorMessage represents the error message describing part of the SQL code that caused the error.
<a href="#">InitGDS</a>	Initializes InterBase client environment.



[LoadedGDS](#)

Returns True when InterBase client library has already been loaded.

[LoadGDS](#)

Loads InterBase client library to memory.

## Variables

Name	Description
<a href="#">GDSDLL</a>	Use GDSDLL to read and assign name of InterBase client library file. By default
<a href="#">GDSVersion</a>	Read GDSVersion to learn version of InterBase client library.
<a href="#">GDSVersionSt</a>	Read GDSVersionSt to learn version of InterBase client library as string.
<a href="#">IBXMLDLL</a>	Use IBXMLDLL to read and assign name of InterBase client library file.

## Constants

Name	Description
<a href="#">DACProductName</a>	Read this constant to get product name of current components.

© 1997-2013 Devart. All Rights Reserved.

## 17.17.1 Routines

Routines in the **IBCCall** unit.

## Routines

Name	Description
<a href="#">DetectGDS</a>	Searches InterBase client library and sets GDSDLL, GDSVersion variables.
<a href="#">FreeGDS</a>	Removes InterBase client library from memory.
GetIBError	Returns error messages for given error code and status vector. ErrorMessage represents the error message InterBase provides for the error. SQLErrorMessage represents the error message describing part of the SQL code that caused the error.

[InitGDS](#)

Initiali es InterBase client enviroment.

[LoadedGDS](#)

Returns True when InterBase client library has already been loaded.

[LoadGDS](#)

Loads InterBase client library to memory.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.17.1.1 DetectGDS Procedure

Searches InterBase client library and sets GDSDLL, GDSVersion variables.

##### Unit

[IBCCall](#)

##### Syntax

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.17.1.2 FreeGDS Procedure

Removes InterBase client library from memory.

##### Unit

[IBCCall](#)

##### Syntax

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.17.1.3 InitGDS Procedure

Initiali es InterBase client enviroment.

##### Unit

[IBCCall](#)

##### Syntax

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.17.1.4 LoadedGDS Procedure

Returns True when InterBase client library has already been loaded.

##### Unit

[IBCCall](#)

##### Syntax

---

© 1997-2013 Devart. All Rights Reserved.

### 17.17.1.5 LoadGDS Procedure

Loads InterBase client library to memory.

#### Unit

[IBCCall](#)

#### Syntax

---

© 1997-2013 Devart. All Rights Reserved.

### 17.17.2 Variables

Variables in the **IBCCall** unit.

#### Variables

Name	Description
<a href="#">GDSDLL</a>	Use GDSDLL to read and assing name of InterBase client library file. By default
<a href="#">GDSVersion</a>	Read GDSVersion to learn version of InterBase client library.
<a href="#">GDSVersionSt</a>	Read GDSVersionSt to learn version of InterBase client library as string.
<a href="#">IBXMLDLL</a>	Use IBXMLDLL to read and assing name of InterBase client library file.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.17.2.1 GDSDLL Variable

Use GDSDLL to read and assing name of InterBase client library file. By default

#### Unit

[IBCCall](#)

#### Syntax

#### Example

For example:

```
GDSDLL := 'C:\WINDOWS\SYSTEM32\GDS32.DLL'
```

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.17.2.2 GDSVersion Variable

Read GDSVersion to learn version of InterBase client library.

##### Unit

[IBCCall](#)

##### Syntax

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.17.2.3 GDSVersionSt Variable

Read GDSVersionSt to learn version of InterBase client library as string.

##### Unit

[IBCCall](#)

##### Syntax

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.17.2.4 IBXMLDLL Variable

Use IBXMLDLL to read and assing name of InterBase client library file.

##### Unit

[IBCCall](#)

##### Syntax

---

© 1997-2013 Devart. All Rights Reserved.

### 17.17.3 Constants

Constants in the **IBCCall** unit.

#### Constants

Name	Description
<a href="#">DACProductName</a>	Read this constant to get product name of current components.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.17.3.1 DACProductName Constant

Read this constant to get product name of current components.

##### Unit

[IBCCall](#)

##### Syntax

---

```
DACProductName = 'IBDAC';
```

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18 IBCClasses

IBCClasses unit defines the following data type constants: dtDbKey dtFixedChar dtFixedWideChar

### Classes

Name	Description
<a href="#">TGSDDatabaseInfo</a>	A class providing information about an InterBase database.
<a href="#">TIBCBlob</a>	A class holding value of the BLOB fields and parameters.

### Enumerations

Name	Description
<a href="#">TIBCIsoIationLevel</a>	Specifies the transaction isolation level and access mode.

### Routines

Name	Description
<a href="#">Reverse2</a>	Switches places of bytes in the word argument.
<a href="#">XSQLDA_LENGTH</a>	Analogue of InterBase XSQLDA_LENGTH macro. Calculates the number of bytes that must be allocated for an input or output XSQLDA.

### Variables

Name	Description
<a href="#">IntegerPrecision</a>	Set this constant to define the type of NUMERIC and DECIMAL fields with precision less or equal than IntegerPrecision as dtInteger. Otherwise, they are defined as dtFloat.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.18.1 Classes

Classes in the **IBCClasses** unit.

#### Classes

Name	Description
<a href="#">TGDSDatabaseInfo</a>	A class providing information about an InterBase database.
<a href="#">TIBCBlob</a>	A class holding value of the BLOB fields and parameters.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1 TGDSDatabaseInfo Class

A class providing information about an InterBase database.  
For a list of all members of this type, see [TGDSDatabaseInfo](#) members.

#### Unit

[IBCClasses](#)

#### Syntax

```
TGDSDatabaseInfo = class (System.TObject);
```

#### Remarks

Use the TGDSDatabaseInfo class to get information about InterBase database.

#### See Also

- [TIBCCConnection.DatabaseInfo](#)
- 

© 1997-2013 Devart. All Rights Reserved.

##### 17.18.1.1.1 Members

[TGDSDatabaseInfo](#) class overview.

#### Properties

Name	Description
<a href="#">Allocation</a>	Indicates the number of database pages allocated.
<a href="#">AttachmentID</a>	Indicates a unique connection identifier.
<a href="#">BackoutCount</a>	Contains the number of removals of a record version.
<a href="#">BaseLevel</a>	Indicates the database version number.

<a href="#"><u>CurLogFileName</u></a>	Contains the name of the log-file.
<a href="#"><u>CurLogPartitionOffset</u></a>	Indicates the value of isc info cur log part offset .
<a href="#"><u>CurrentMemory</u></a>	Indicates the amount of the server memory currently in use.
<a href="#"><u>DBFileName</u></a>	Contains the database filename.
<a href="#"><u>DBImplementationClass</u></a>	Indicates the database implementation class number.
<a href="#"><u>DBImplementationNo</u></a>	Indicates the database implementation number.
<a href="#"><u>DBSiteName</u></a>	Contains the database site name.
<a href="#"><u>DeleteCount</u></a>	Holds the number of database deletes since the database was last attached.
<a href="#"><u>ExpungeCount</u></a>	Indicates the number of removals of a record and all of its ancestors.
<a href="#"><u>Fetches</u></a>	Indicates the number of reads from the memory buffer cache.
<a href="#"><u>ForcedWrites</u></a>	Used to indicate the mode in which database writes are performed.
<a href="#"><u>InsertCount</u></a>	Holds the number of insertions into the database since the database was last attached.
<a href="#"><u>IsRemoteConnect</u></a>	Used to indicate whether the connection with the database is remote.
<a href="#"><u>LogFile</u></a>	Used to indicate the value of isc info log file.
<a href="#"><u>Marks</u></a>	Used to indicate the number of writes to the memory buffer cache.
<a href="#"><u>MaxMemory</u></a>	Indicates the maximum amount of memory used at one time since the first process attached to the database.

<a href="#"><u>NoReserve</u></a>	Specifies if the space for holding backup versions of modified records is reserved on each database page.
<a href="#"><u>NumBuffers</u></a>	Indicates the number of currently allocated memory buffers.
<a href="#"><u>NumWALBuffers</u></a>	Indicates the value of isc info num wal buffers.
<a href="#"><u>ODSMajorVersion</u></a>	Indicates the on disk structure (ODS) major version number.
<a href="#"><u>ODSMinorVersion</u></a>	Indicates the on disk structure (ODS) minor version number.
<a href="#"><u>PageSize</u></a>	Shows the number of bytes per page for the database.
<a href="#"><u>PurgeCount</u></a>	Holds the number of removals of records committed from the database, resulting in older versions.
<a href="#"><u>ReadIdxCount</u></a>	Holds the number of reads done via an index since the database was last attached.
<a href="#"><u>ReadOnly</u></a>	Indicates whether the database is read only.
<a href="#"><u>Reads</u></a>	Indicates the number of page reads from the database since the current database was first attached.
<a href="#"><u>ReadSeqCount</u></a>	Contains the number of sequential database reads done on each table since the database was last attached.
<a href="#"><u>SweepInterval</u></a>	Indicates the number of transactions that are committed between "sweeps".
<a href="#"><u>UpdateCount</u></a>	Contains the number of database updates since the database was last attached.
<a href="#"><u>UserNames</u></a>	Contains the names of all users currently attached to the database.
<a href="#"><u>Version</u></a>	Indicates the version of the database implementation.
<a href="#"><u>WALAverageGroupCommitSize</u></a>	Indicates the value of isc info wal avg grpc size.



<a href="#">WALAverageIOSize</a>	Indicates the value of isc info wal avg io size.
<a href="#">WALBufferSize</a>	Indicates the value of isc info wal buffer size.
<a href="#">WALCheckpointLength</a>	Indicates the value of isc info wal ckpt length.
<a href="#">WALCurCheckpointInterval</a>	Indicates the value of isc info wal cur ckpt interval.
<a href="#">WALGroupCommitWaitUSecs</a>	Indicates the value of isc info wal grpc wait usecs.
<a href="#">WALNumCommits</a>	Indicates the value of isc info wal num commits.
<a href="#">WALNumIO</a>	Indicates the value of isc info wal num id.
<a href="#">WALPrvCheckpointFilename</a>	Indicates the value of isc info wal prv ckpt filename.
<a href="#">WALPrvCheckpointPartOffset</a>	Indicates the value of isc info wal prv ckpt poffset.
<a href="#">Writes</a>	Indicates the number of page writes to the current database since it was first attached by any process.

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2 Properties

Properties of the **TGDSDatabaseInfo** class.  
For a complete list of the **TGDSDatabaseInfo** class members, see the  
[TGDSDatabaseInfo Members](#) topic.

#### Public

Name	Description
<a href="#">Allocation</a>	Indicates the number of database pages allocated.
<a href="#">AttachmentID</a>	Indicates a unique connection identifier.
<a href="#">BackoutCount</a>	Contains the number of removals of a record version.
<a href="#">BaseLevel</a>	Indicates the database version number.
<a href="#">CurLogFileName</a>	Contains the name of the log-file.

<a href="#"><u>CurLogPartitionOffset</u></a>	Indicates the value of isc info cur log part offset .
<a href="#"><u>CurrentMemory</u></a>	Indicates the amount of the server memory currently in use.
<a href="#"><u>DBFileName</u></a>	Contains the database filename.
<a href="#"><u>DBImplementationClass</u></a>	Indicates the database implementation class number.
<a href="#"><u>DBImplementationNo</u></a>	Indicates the database implementation number.
<a href="#"><u>DBSiteName</u></a>	Contains the database site name.
<a href="#"><u>DeleteCount</u></a>	Holds the number of database deletes since the database was last attached.
<a href="#"><u>ExpungeCount</u></a>	Indicates the number of removals of a record and all of its ancestors.
<a href="#"><u>Fetches</u></a>	Indicates the number of reads from the memory buffer cache.
<a href="#"><u>ForcedWrites</u></a>	Used to indicate the mode in which database writes are performed.
<a href="#"><u>InsertCount</u></a>	Holds the number of insertions into the database since the database was last attached.
<a href="#"><u>IsRemoteConnect</u></a>	Used to indicate whether the connection with the database is remote.
<a href="#"><u>LogFile</u></a>	Used to indicate the value of isc info log file.
<a href="#"><u>Marks</u></a>	Used to indicate the number of writes to the memory buffer cache.
<a href="#"><u>MaxMemory</u></a>	Indicates the maximum amount of memory used at one time since the first process attached to the database.
<a href="#"><u>NoReserve</u></a>	Specifies if the space for holding backup versions of modified records is reserved on each database page.

<a href="#"><u>NumBuffers</u></a>	Indicates the number of currently allocated memory buffers.
<a href="#"><u>NumWALBuffers</u></a>	Indicates the value of isc info num wal buffers.
<a href="#"><u>ODSMajorVersion</u></a>	Indicates the on disk structure (ODS) major version number.
<a href="#"><u>ODSMinorVersion</u></a>	Indicates the on disk structure (ODS) minor version number.
<a href="#"><u>PageSize</u></a>	Shows the number of bytes per page for the database.
<a href="#"><u>PurgeCount</u></a>	Holds the number of removals of records committed from the database, resulting in older versions.
<a href="#"><u>ReadIdxCount</u></a>	Holds the number of reads done via an index since the database was last attached.
<a href="#"><u>ReadOnly</u></a>	Indicates whether the database is read only.
<a href="#"><u>Reads</u></a>	Indicates the number of page reads from the database since the current database was first attached.
<a href="#"><u>ReadSeqCount</u></a>	Contains the number of sequential database reads done on each table since the database was last attached.
<a href="#"><u>SweepInterval</u></a>	Indicates the number of transactions that are committed between "sweeps".
<a href="#"><u>UpdateCount</u></a>	Contains the number of database updates since the database was last attached.
<a href="#"><u>UserNames</u></a>	Contains the names of all users currently attached to the database.
<a href="#"><u>Version</u></a>	Indicates the version of the database implementation.
<a href="#"><u>WALAverageGroupCommitSize</u></a>	Indicates the value of isc info wal avg grpc size.
<a href="#"><u>WALAverageIOSize</u></a>	Indicates the value of isc info wal avg io size.
<a href="#"><u>WALBufferSize</u></a>	Indicates the value of isc info wal buffer size.

[WALCheckpointLength](#)

Indicates the value of  
isc info wal ckpt length.

[WALCurCheckpointInterval](#)

Indicates the value of  
isc info wal cur ckpt inter  
val.

[WALGroupCommitWaitUSecs](#)

Indicates the value of  
isc info wal grpc wait usec  
s.

[WALNumCommits](#)

Indicates the value of  
isc info wal num commits.

[WALNumIO](#)

Indicates the value of  
isc info wal num id.

[WALPrvCheckpointFilename](#)

Indicates the value of  
isc info wal prv ckpt fnam  
e.

[WALPrvCheckpointPartOffset](#)

Indicates the value of  
isc info wal prv ckpt poffs  
et.

[Writes](#)

Indicates the number of  
page writes to the current  
database since it was first  
attached by any process.

**See Also**

- [TGDSDatabaseInfo Class](#)
  - [TGDSDatabaseInfo Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.1 Allocation Property

Indicates the number of database pages allocated.

**Class**[TGDSDatabaseInfo](#)**Syntax**

```
property Allocation: integer;
```

**Remarks**

Use the Allocation property to determine the number of database pages allocated.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.2 AttachmentID Property

Indicates a unique connection identifier.

**Class**[TGDSDatabaseInfo](#)

**Syntax**

```
property AttachmentID: integer;
```

**Remarks**

Use the AttachmentID property to indicate a unique connection identifier.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.3 BackoutCount Property

Contains the number of removals of a record version.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property BackoutCount: TStringList;
```

**Remarks**

Use the BackoutCount property to indicate the number of removals of a record version.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.4 BaseLevel Property

Indicates the database version number.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property BaseLevel: integer;
```

**Remarks**

Use the BaseLevel property to determine the database version number.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.5 CurLogFileName Property

Contains the name of the log-file.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property CurLogFileName: string;
```

**Remarks**

Contains the name of the log-file. The TIBCCConnection component must be active, otherwise an exception is raised.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.6 CurLogPartitionOffset Property

Indicates the value of isc info cur log part offset.

##### **Class**

[TGDSDatabaseInfo](#)

##### **Syntax**

```
property CurLogPartitionOffset: integer;
```

##### **Remarks**

Use the CurLogPartitionOffset property to determine the value of isc info cur log part offset.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.7 CurrentMemory Property

Indicates the amount of the server memory currently in use.

##### **Class**

[TGDSDatabaseInfo](#)

##### **Syntax**

```
property CurrentMemory: integer;
```

##### **Remarks**

Use the CurrentMemory property to determine the amount of server memory (in bytes) currently in use.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.8 DBFileName Property

Contains the database filename.

##### **Class**

[TGDSDatabaseInfo](#)

##### **Syntax**

```
property DBFileName: string;
```

##### **Remarks**

Contains the database filename.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.9 DBImplementationClass Property

Indicates the database implementation class number.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property DBImplementationClass: integer;
```

**Remarks**

Use the DBImplementationClass property to determine the database implementation class number.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.10 DBImplementationNo Property

Indicates the database implementation number.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property DBImplementationNo: integer;
```

**Remarks**

Use the DBImplementationNo to determine the database implementation number.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.11 DBSiteName Property

Contains the database site name.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property DBSiteName: string;
```

**Remarks**

Contains the database site name.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.12 DeleteCount Property

Holds the number of database deletes since the database was last attached.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property DeleteCount: TStringList;
```

**Remarks**

Contains the number of database deletes since the database was last attached.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.13 ExpungeCount Property

Indicates the number of removals of a record and all of its ancestors.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property ExpungeCount: TStringList;
```

**Remarks**

Use the ExpungeCount property to determine the number of removals of a record and all of its ancestors.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.14 Fetches Property

Indicates the number of reads from the memory buffer cache.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property Fetches: integer;
```

**Remarks**

Use the Fetches property to determine the number of reads from the memory buffer cache.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.15 ForcedWrites Property

Used to indicate the mode in which database writes are performed.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property ForcedWrites: integer;
```

**Remarks**



---

Use the FoprcedWrites property to indicate the mode in which database writes are performed. It is 0 for asynchronous mode, 1 for synchronous mode

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.16 InsertCount Property

Holds the number of insertions into the database since the database was last attached.

#### Class

[TGDSDatabaseInfo](#)

#### Syntax

```
property InsertCount: TStringList;
```

#### Remarks

Contains the number of insertions into the database since the database was last attached.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.17 IsRemoteConnect Property

Used to indicate whether the connection with the database is remote.

#### Class

[TGDSDatabaseInfo](#)

#### Syntax

```
property IsRemoteConnect: boolean;
```

#### Remarks

Use the IsRemoteConnect property to determine whether the connection with the database is remote.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.18 LogFile Property

Used to indicate the value of isc info log file.

#### Class

[TGDSDatabaseInfo](#)

#### Syntax

```
property LogFile: integer;
```

#### Remarks

Use the LogFile property to indicate the value of isc info log file.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.19 Marks Property

Used to indicate the number of writes to the memory buffer cache.

##### **Class**

[TGDSDatabaseInfo](#)

##### **Syntax**

```
property Marks: integer;
```

##### **Remarks**

Use the Marks property to determine the number of writes to the memory buffer cache.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.20 MaxMemory Property

Indicates the maximum amount of memory used at one time since the first process attached to the database.

##### **Class**

[TGDSDatabaseInfo](#)

##### **Syntax**

```
property MaxMemory: integer;
```

##### **Remarks**

Use the MaxMemory property to indicate the maximum amount of memory used at one time since the first process attached to the database.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.21 NoReserve Property

Specifies if the space for holding backup versions of modified records is reserved on each database page.

##### **Class**

[TGDSDatabaseInfo](#)

##### **Syntax**

```
property NoReserve: integer;
```

##### **Remarks**

If 0, the space is reserved on each database page for holding backup versions of modified records.

If 1, no space is reserved for such records.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.22 NumBuffers Property

Indicates the number of currently allocated memory buffers.

##### **Class**

[TGDSDatabaseInfo](#)

##### **Syntax**

```
property NumBuffers: integer;
```

##### **Remarks**

Use the NumBuffers property to indicate the number of currently allocated memory buffers.

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.23 NumWALBuffers Property

Indicates the value of isc info num wal buffers.

##### **Class**

[TGDSDatabaseInfo](#)

##### **Syntax**

```
property NumWALBuffers: integer;
```

##### **Remarks**

Use the NumWALBuffers property to indicate the value of isc info num wal buffers.

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.24 ODSMajorVersion Property

Indicates the on disk structure (ODS) major version number.

##### **Class**

[TGDSDatabaseInfo](#)

##### **Syntax**

```
property ODSMajorVersion: integer;
```

##### **Remarks**

Use the ODSMajorVersion to determine the on disk structure (ODS) major version number.

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.25 ODSMinorVersion Property

Indicates the on disk structure (ODS) minor version number.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property ODSMinorVersion: integer;
```

**Remarks**

Use the ODSMinorVersion property to determine the on disk structure (ODS) minor version number.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.26 PageSize Property

Shows the number of bytes per page for the database.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property PageSize: integer;
```

**Remarks**

Shows the number of bytes per page for the database.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.27 PurgeCount Property

Holds the number of removals of records committed from the database, resulting in older versions.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property PurgeCount: TStringList;
```

**Remarks**

Contains the number of removals of records committed from the database, resulting in older versions.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.28 ReadIdxCount Property

Holds the number of reads done via an index since the database was last attached.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property ReadIdxCount: TStringList;
```

**Remarks**

The ReadIdxCount property is used to contain the number of reads done via an index since the database was last attached.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.29 ReadOnly Property

Indicates whether the database is read only.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property ReadOnly: Boolean;
```

**Remarks**

The ReadOnly property is used to indicate whether the database is read only.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.30 Reads Property

Indicates the number of page reads from the database since the current database was first attached.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property Reads: integer;
```

**Remarks**

The Reads property is used to indicate the number of page reads from the database since the current database was first attached.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.31 ReadSeqCount Property

Contains the number of sequential database reads done on each table since the database was last attached.

**Class**[TGDSDatabaseInfo](#)**Syntax****property** ReadSeqCount: TStringList;**Remarks**

The ReadSeqCount property is used to contain the number of sequential database reads done on each table since the database was last attached.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.32 SweepInterval Property

Indicates the number of transactions that are committed between "sweeps".

**Class**[TGDSDatabaseInfo](#)**Syntax****property** SweepInterval: integer;**Remarks**

Use the SweepInterval property to indicate the number of transactions that are committed between "sweeps".

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.33 UpdateCount Property

Contains the number of database updates since the database was last attached.

**Class**[TGDSDatabaseInfo](#)**Syntax****property** UpdateCount: TStringList;**Remarks**

The UpdateCount property is used to hold the number of database updates since the database was last attached.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.34 UserNames Property

Contains the names of all users currently attached to the database.

**Class**[TGDSDatabaseInfo](#)

**Syntax**

```
property UserNames: TStringList;
```

**Remarks**

The UserNames property is used to hold the names of all users currently attached to the database.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.35 Version Property

Indicates the version of the database implementation.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property Version: string;
```

**Remarks**

The Version property is used to indicate the version of the database implementation.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.36 WALAverageGroupCommitSize Property

Indicates the value of isc info wal avg grpc si e.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property WALAverageGroupCommitSize: integer;
```

**Remarks**

The WALAverageGroupCommitSize property is used to indicate the value of isc info wal avg grpc si e.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.37 WALAverageIOSize Property

Indicates the value of isc info wal avg io si e.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property WALAverageIOSize: integer;
```

**Remarks**

The WALAverageIOSize property is used to indicate the value of isc info wal avg io size.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.38 WALBufferSize Property

Indicates the value of isc info wal buffer size.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property WALBufferSize: integer;
```

**Remarks**

The WALBufferSize property is used to indicate the value of isc info wal buffer size.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.39 WALCheckpointLength Property

Indicates the value of isc info wal ckpt length.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property WALCheckpointLength: integer;
```

**Remarks**

The WALCheckpointLength is used to indicate the value of isc info wal ckpt length.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.40 WALCurCheckpointInterval Property

Indicates the value of isc info wal cur ckpt interval.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property WALCurCheckpointInterval: integer;
```

**Remarks**

The WALCurCheckpointInterval property is used to indicate the value of



---

isc info wal cur ckpt interval.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.41 WALGroupCommitWaitUSecs Property

Indicates the value of isc info wal grpc wait usecs.

##### **Class**

[TGDSDatabaseInfo](#)

##### **Syntax**

```
property WALGroupCommitWaitUSecs: integer;
```

##### **Remarks**

The WALGroupCommitWaitUSec property is used to indicate the value of isc info wal grpc wait usecs.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.42 WALNumCommits Property

Indicates the value of isc info wal num commits.

##### **Class**

[TGDSDatabaseInfo](#)

##### **Syntax**

```
property WALNumCommits: integer;
```

##### **Remarks**

The WALNumCommits property is used to indicate the value of isc info wal num commits.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.1.2.43 WALNumIO Property

Indicates the value of isc info wal num id.

##### **Class**

[TGDSDatabaseInfo](#)

##### **Syntax**

```
property WALNumIO: integer;
```

##### **Remarks**

The WALNumI property is used to indicate the value of isc info wal num id.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.44 WALPrvCheckpointFilename Property

Indicates the value of isc info wal prv ckpt fname.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property WALPrvCheckpointFilename: string;
```

**Remarks**

The WALPrvCheckpointFilename property is used to indicate the value of isc info wal prv ckpt fname.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.45 WALPrvCheckpointPartOffset Property

Indicates the value of isc info wal prv ckpt poffset.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property WALPrvCheckpointPartOffset: integer;
```

**Remarks**

The WALPrvCheckpointPartOffset property is used to indicate the value of isc info wal prv ckpt poffset.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.1.2.46 Writes Property

Indicates the number of page writes to the current database since it was first attached by any process.

**Class**

[TGDSDatabaseInfo](#)

**Syntax**

```
property Writes: integer;
```

**Remarks**

The Writes property is used to indicate the number of page writes to the current database since it was first attached by any process.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.18.1.2 TIBCBlob Class

A class holding value of the BLOB fields and parameters.  
For a list of all members of this type, see [TIBCBlob](#) members.

#### Unit

[IBCClasses](#)

#### Syntax

```
TIBCBlob = class (TCompressedBlob) ;
```

#### Remarks

TIBCBlob is a descendant of TCompressedBlob class. It holds value of the BLOB fields and parameters.

#### Inheritance Hierarchy

[TSharedObject](#)  
[TBlob](#)  
[TCompressedBlob](#)  
**TIBCBlob**

#### See Also

- [TCompressedBlob](#)
- [TCustomIBCDDataSet.GetBlob](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.2.1 Members

[TIBCBlob](#) class overview.

#### Properties

Name	Description
<a href="#">AsString</a> (inherited from <a href="#">TBlob</a> )	Used to manipulate BLOB value as string.
<a href="#">AsWideString</a> (inherited from <a href="#">TBlob</a> )	Used to manipulate BLOB value as Unicode string.
<a href="#">Cached</a>	Indicates whether the BLOB data are cached on the client or they are accessed remotely on the server.
<a href="#">CharsetID</a>	Source charset for BLOB subtype conversion using BLOB filters.
<a href="#">Compressed</a> (inherited from <a href="#">TCompressedBlob</a> )	Used to indicate if the Blob is compressed.
<a href="#">CompressedSize</a> (inherited from <a href="#">TCompressedBlob</a> )	Used to indicate compressed size of the Blob data.

[ConversionCharsetID](#)

Target charset for BLOB subtype conversion using Blob filters.

[ConversionSubType](#)

Target subtype for the BLOB subtype conversion using BLOB filters.

[DbHandle](#)

Contains the handle of the database where the BLOB is stored.

[Handle](#)

Contains the BLOB handle.

[ID](#)

Contains BLOB ID.

[IsUnicode](#) (inherited from [TBlob](#))

Gives choice of making TBlob store and process data in Unicode format or not.

[MaxSegmentSize](#)

Indicates the largest BLOB segment size.

[NumSegments](#)

Indicates the number of the BLOB segments in the database.

[RefCount](#) (inherited from [TSharedObject](#))

Used to return the count of reference to a TSharedObject object.

[Size](#) (inherited from [TBlob](#))

Used to learn the size of the TBlob value in bytes.

[Streamed](#)

Indicates whether the BLOB is stream or segmented.

[SubType](#)

Source subtype for the BLOB subtype conversion using BLOB filters.

[TrHandle](#)

Contains the handle of the transaction in which the BLOB is read or written.

## Methods

### Name

### Description

[AddRef](#) (inherited from [TSharedObject](#))

Increments the reference count for the number of references dependent on the TSharedObject object.

[AllocBlob](#)

Allocates and initializes the Blob handle.

[Assign](#) (inherited from [TBlob](#))

Sets BLOB value from another TBlob object.

[Clear](#) (inherited from [TBlob](#))

Deletes the current value in TBlob object.

[CloseBlob](#)

Frees the Blob handle.

[FreeBlob](#)

Clears BLOB ID.

<a href="#">IsInit</a>	Verifies BLOB ID initiali ation.
<a href="#">LengthBlob</a>	Determines the number of bytes contained in the Blob object.
<a href="#">LoadFromFile</a> (inherited from <a href="#">TBlob</a> )	Loads the contents of a file into a TBlob object.
<a href="#">LoadFromStream</a> (inherited from <a href="#">TBlob</a> )	Copies the contents of a stream into the TBlob object.
<a href="#">Read</a> (inherited from <a href="#">TBlob</a> )	Acquires a raw sequence of bytes from the data stored in TBlob.
<a href="#">ReadBlob</a>	Reads BLOB from the database.
<a href="#">Release</a> (inherited from <a href="#">TSharedObject</a> )	Decrements the reference count.
<a href="#">SaveToFile</a> (inherited from <a href="#">TBlob</a> )	Saves the contents of the TBlob object to a file.
<a href="#">SaveToStream</a> (inherited from <a href="#">TBlob</a> )	Copies the contents of a TBlob object to a stream.
<a href="#">Truncate</a> (inherited from <a href="#">TBlob</a> )	Sets new TBlob si e and discards all data over it.
<a href="#">Write</a> (inherited from <a href="#">TBlob</a> )	Stores a raw sequence of bytes into a TBlob object.
<a href="#">WriteBlob</a>	Writes BLOB to the database.

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.2.2 Properties

Properties of the **TIBCBlob** class.

For a complete list of the **TIBCBlob** class members, see the [TIBCBlob Members](#) topic.

#### Public

Name	Description
<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">Assign</a> (inherited from <a href="#">TBlob</a> )	Sets BLOB value from another TBlob object.
<a href="#">AsString</a> (inherited from <a href="#">TBlob</a> )	Used to manipulate BLOB value as string.
<a href="#">AsWideString</a> (inherited from <a href="#">TBlob</a> )	Used to manipulate BLOB value as Unicode string.

<a href="#">Cached</a>	Indicates whether the BLOB data are cached on the client or they are accessed remotely on the server.
<a href="#">CharsetID</a>	Source charset for BLOB subtype conversion using BLOB filters.
<a href="#">Clear</a> (inherited from <a href="#">TBlob</a> )	Deletes the current value in TBlob object.
<a href="#">Compressed</a> (inherited from <a href="#">TCompressedBlob</a> )	Used to indicate if the Blob is compressed.
<a href="#">CompressedSize</a> (inherited from <a href="#">TCompressedBlob</a> )	Used to indicate compressed size of the Blob data.
<a href="#">ConversionCharsetID</a>	Target charset for BLOB subtype conversion using Blob filters.
<a href="#">ConversionSubType</a>	Target subtype for the BLOB subtype conversion using BLOB filters.
<a href="#">DbHandle</a>	Contains the handle of the database where the BLOB is stored.
<a href="#">Handle</a>	Contains the BLOB handle.
<a href="#">ID</a>	Contains BLOB ID.
<a href="#">IsUnicode</a> (inherited from <a href="#">TBlob</a> )	Gives choice of making TBlob store and process data in Unicode format or not.
<a href="#">LoadFromFile</a> (inherited from <a href="#">TBlob</a> )	Loads the contents of a file into a TBlob object.
<a href="#">LoadFromStream</a> (inherited from <a href="#">TBlob</a> )	Copies the contents of a stream into the TBlob object.
<a href="#">MaxSegmentSize</a>	Indicates the largest BLOB segment size.
<a href="#">NumSegments</a>	Indicates the number of the BLOB segments in the database.
<a href="#">Read</a> (inherited from <a href="#">TBlob</a> )	Acquires a raw sequence of bytes from the data stored in TBlob.
<a href="#">RefCount</a> (inherited from <a href="#">TSharedObject</a> )	Used to return the count of reference to a TSharedObject object.
<a href="#">Release</a> (inherited from <a href="#">TSharedObject</a> )	Decrements the reference count.
<a href="#">SaveToFile</a> (inherited from <a href="#">TBlob</a> )	Saves the contents of the TBlob object to a file.
<a href="#">SaveToStream</a> (inherited from <a href="#">TBlob</a> )	Copies the contents of a TBlob object to a stream.

[Size](#) (inherited from [TBlob](#))

Used to learn the size of the TBlob value in bytes.

[Streamed](#)

Indicates whether the BLOB is stream or segmented.

[SubType](#)

Source subtype for the BLOB subtype conversion using BLOB filters.

[TrHandle](#)

Contains the handle of the transaction in which the BLOB is read or written.

[Truncate](#) (inherited from [TBlob](#))

Sets new TBlob size and discards all data over it.

[Write](#) (inherited from [TBlob](#))

Stores a raw sequence of bytes into a TBlob object.

### See Also

- [TIBCBlob Class](#)
- [TIBCBlob Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.2.2.1 Cached Property

Indicates whether the BLOB data are cached on the client or they are accessed remotely on the server.

### Class

[TIBCBlob](#)

### Syntax

```
property Cached: boolean;
```

### Remarks

The Cached property is used to indicate whether the BLOB data are cached on the client or they are accessed remotely on the server. In most cases you don't need to set the value of this property directly. To enable or disable BLOB caching, use the [TCustomIBCDDataSet.Options](#) property.

### See Also

- [TCustomIBCDDataSet.Options](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.2.2.2 CharSetID Property

Source charset for BLOB subtype conversion using BLOB filters.

### Class

[TIBCBlob](#)

**Syntax**

```
property CharsetID: integer;
```

**Remarks**

Source charset for BLOB subtype conversion using BLOB filters. It corresponds to isc bpb source interp parameter in BLOB Parameter Buffer (BPB) that is used for filtration. For more information on BLOB filters refer to InterBase API Guide.

**See Also**

- [ConversionCharsetID](#)
  - [SubType](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.2.2.3 ConversionCharsetID Property

Target charset for BLOB subtype conversion using Blob filters.

**Class**

[TIBCBlob](#)

**Syntax**

```
property ConversionCharsetID: integer;
```

**Remarks**

Target charset for BLOB subtype conversion using Blob filters. It corresponds to isc bpb target interp parameter in BLOB Parameter Buffer (BPB) that is used for filtration. For more information on BLOB filters refer to InterBase API Guide.  
Set ConversionCharset property to the charset that you want to get when reading the BLOB from database.  
Set ConversionCharset property to the charset that you want BLOB to store to database in when writing BLOB to database.

**See Also**

- [CharsetID](#)
  - [ConversionSubType](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.2.2.4 ConversionSubType Property

Target subtype for the BLOB subtype conversion using BLOB filters.

**Class**

[TIBCBlob](#)



## Syntax

**property** ConversionSubType: integer;

## Remarks

Target subtype for the BLOB subtype conversion using BLOB filters. It corresponds to the *isc bpb target type* parameter in BLOB Parameter Buffer (BPB) that is used for filtration. For more information about Blob subtypes and filters refer to InterBase API Guide.

Set the ConversionSubType property to the Blob subtype that you want to get when reading the Blob from database.

Set the ConversionSubType property to the Blob subtype that you want to store to database when writing Blob to database.

## See Also

- [SubType](#)
- [ConversionCharsetID](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.18.1.2.2.5 DbHandle Property

Contains the handle of the database where the BLOB is stored.

## Class

[TIBCBlob](#)

## Syntax

**property** DbHandle: TISC\_DB\_HANDLE;

## Remarks

The DbHandle property is used to contain the handle of the database where the BLOB is stored.

## See Also

- [TIBConnection.Handle](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.18.1.2.2.6 Handle Property

Contains the BLOB handle.

## Class

[TIBCBlob](#)

## Syntax

```
property Handle: TISC_BLOB_HANDLE;
```

**Remarks**

Contains the BLOB handle. Use the Handle property for direct calls to InterBase BLOB API.

**See Also**

- [AllocBlob](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.2.2.7 ID Property

Contains BLOB ID.

**Class**

[TIBCBlob](#)

**Syntax**

```
property ID: TISC_QUAD;
```

**Remarks**

Contains BLOB ID that is actually stored in the BLOB field of the table record. It is a unique numeric value that references the BLOB data.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.2.2.8 MaxSegmentSize Property

Indicates the largest BLOB segment size.

**Class**

[TIBCBlob](#)

**Syntax**

```
property MaxSegmentSize: Word;
```

**Remarks**

The MaxSegmentSize property is used to indicate the size in bytes of the largest segment of the BLOB.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.2.2.9 NumSegments Property

Indicates the number of the BLOB segments in the database.

**Class**

[TIBCBlob](#)

## Syntax

**property** NumSegments: Cardinal;

## Remarks

Use the NumSegments property to indicate the number of the BLOB segments in the database.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.18.1.2.2.10 Streamed Property

Indicates whether the BLOB is stream or segmented.

## Class

[TIBCBlob](#)

## Syntax

**property** Streamed: boolean;

## Remarks

Use the Streamed property to determine whether the BLOB is stream or segmented. Segmented BLOBs are usual InterBase BLOBs and are stored in chunks. Stream BLOBs are stored as a continuous array of data bytes.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.18.1.2.2.11 SubType Property

Source subtype for the BLOB subtype conversion using BLOB filters.

## Class

[TIBCBlob](#)

## Syntax

**property** SubType: integer;

## Remarks

Source subtype for the BLOB subtype conversion using BLOB filters. It corresponds to *isc bpb source type* parameter in BLOB Parameter Buffer (BPB) that is used for filtration. For more information on BLOB subtypes and filters refer to InterBase API Guide.

## See Also

- [ConversionSubType](#)
- [CharsetID](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.2.2.12 TrHandle Property

Contains the handle of the transaction in which the BLOB is read or written.

**Class**

[TIBCBlob](#)

**Syntax**

```
property TrHandle: TISC_TR_HANDLE;
```

**Remarks**

Use the TrHandle property to hold the handle of the transaction in which the BLOB is read or written.

**See Also**

- [TIBCTransaction.Handle](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.2.3 Methods

Methods of the **TIBCBlob** class.

For a complete list of the **TIBCBlob** class members, see the [TIBCBlob Members](#) topic.

**Public**

Name	Description
<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">AllocBlob</a>	Allocates and initializes the Blob handle.
<a href="#">Assign</a> (inherited from <a href="#">TBlob</a> )	Sets BLOB value from another TBlob object.
<a href="#">AsString</a> (inherited from <a href="#">TBlob</a> )	Used to manipulate BLOB value as string.
<a href="#">AsWideString</a> (inherited from <a href="#">TBlob</a> )	Used to manipulate BLOB value as Unicode string.
<a href="#">Clear</a> (inherited from <a href="#">TBlob</a> )	Deletes the current value in TBlob object.
<a href="#">CloseBlob</a>	Frees the Blob handle.
<a href="#">Compressed</a> (inherited from <a href="#">TCompressedBlob</a> )	Used to indicate if the Blob is compressed.
<a href="#">CompressedSize</a> (inherited from <a href="#">TCompressedBlob</a> )	Used to indicate compressed size of the Blob data.
<a href="#">FreeBlob</a>	Clears BLOB ID.

[IsInit](#)

[IsUnicode](#) (inherited from [TBlob](#))

[LengthBlob](#)

[LoadFromFile](#) (inherited from [TBlob](#))

[LoadFromStream](#) (inherited from [TBlob](#))

[Read](#) (inherited from [TBlob](#))

[ReadBlob](#)

[RefCount](#) (inherited from [TSharedObject](#))

[Release](#) (inherited from [TSharedObject](#))

[SaveToFile](#) (inherited from [TBlob](#))

[SaveToStream](#) (inherited from [TBlob](#))

[Size](#) (inherited from [TBlob](#))

[Truncate](#) (inherited from [TBlob](#))

[Write](#) (inherited from [TBlob](#))

[WriteBlob](#)

Verifies BLOB ID initiali ation.

Gives choice of making TBlob store and process data in Unicode format or not.

Determines the number of bytes contained in the Blob object.

Loads the contents of a file into a TBlob object.

Copies the contents of a stream into the TBlob object.

Acquires a raw sequence of bytes from the data stored in TBlob.

Reads BLOB from the database.

Used to return the count of reference to a TSharedObject object.

Decrements the reference count.

Saves the contents of the TBlob object to a file.

Copies the contents of a TBlob object to a stream.

Used to learn the si e of the TBlob value in bytes.

Sets new TBlob si e and discards all data over it.

Stores a raw sequence of bytes into a TBlob object.

Writes BLOB to the database.

## See Also

- [TIBCBlob Class](#)
- [TIBCBlob Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

17.18.1.2.3.1 AllocBlob Method

Allocates and initiali es the Blob handle.

## Class

[TIBCBlob](#)

## Syntax

```
procedure AllocBlob(Read: boolean = True);
```

### Parameters

*Read*

True, if BLOB is opened for reading or writing.

### Remarks

Call the AllocBLOB method to allocate and initialize the Blob handle. The Read parameter indicates whether the BLOB is opened for reading or writing. If the BLOB is opened for writing, new BLOB ID is generated.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.2.3.2 CloseBlob Method

Frees the Blob handle.

### Class

[TIBCBlob](#)

### Syntax

```
procedure CloseBlob;
```

### Remarks

Use the CloseBLOB method to free the Blob handle.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.2.3.3 FreeBlob Method

Clears BLOB ID.

### Class

[TIBCBlob](#)

### Syntax

```
procedure FreeBlob; override;
```

### Remarks

Call the FreeBLOB method to clear BLOB ID.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.1.2.3.4 IsInit Method

Verifies BLOB ID initialization.

### Class

[TIBCBlob](#)

### Syntax

```
function IsInit: boolean;
```

---

**Return Value**

True, if BLOB ID is initialized. False otherwise.

**Remarks**

The IsInit method verifies that BLOB ID is initialized.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.2.3.5 LengthBlob Method

Determines the number of bytes contained in the Blob object.

**Class**

[TIBCBlob](#)

**Syntax**

```
function LengthBlob: longint;
```

**Return Value**

The number of bytes contained in the Blob object.

**Remarks**

The LengthBLOB method returns the number of bytes contained in the Blob object.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.2.3.6 ReadBlob Method

Reads BLOB from the database.

**Class**

[TIBCBlob](#)

**Syntax**

```
procedure ReadBlob;
```

**Remarks**

Call the ReadBLOB method to read BLOB from the database.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.18.1.2.3.7 WriteBlob Method

Writes BLOB to the database.

**Class**

[TIBCBlob](#)

**Syntax**

```
procedure WriteBlob;
```

**Remarks**

Call the WriteBLOB method to write BLOB to the database.

---

© 1997-2013 Devart. All Rights Reserved.

**17.18.2 Enumerations**

Enumerations in the **IBCClasses** unit.

**Enumerations**

Name	Description
<a href="#">TIBCIsoIationLevel</a>	Specifies the transaction isolation level and access mode.

---

© 1997-2013 Devart. All Rights Reserved.

**17.18.2.1 TIBCIsoIationLevel Enumeration**

Specifies the transaction isolation level and access mode.

**Unit**

[IBCClasses](#)

**Syntax**

```
TIBCIsoIationLevel = (ibISnapshot, ibIReadCommitted,  
    ibIReadOnlyReadCommitted, ibITableStability,  
    ibIReadOnlyTableStability, ibICustom);
```

**Values**

Value	Meaning
<b>ibICustom</b>	The parameters of the transaction are set manually in the Params property.
<b>ibIReadCommitted</b>	Enables the transaction to see all committed data in the database, and to update rows updated and committed by other simultaneous transactions without causing lost update problems.
<b>ibIReadOnlyReadCommitted</b>	Enables the transaction to see all committed data in the database with read-only access mode.
<b>ibIReadOnlyTableStability</b>	Provides a transaction read-only access to the tables it uses. Other simultaneous transactions may be able to select rows from these tables, but they can not insert, update, and delete rows from these tables.



<b>ibISnapshot</b>	The default isolation level. Provides a stable, committed view of the database at the time the transaction starts. Other simultaneous transactions can UPDATE and INSERT rows, but this transaction cannot see these changes. For updated rows, this transaction sees versions of these rows as they existed at the start of the transaction. If this transaction attempts to update or delete rows changed by another transaction, an update conflict is reported.
<b>ibITableStability</b>	Provides a transaction sole insert, update, and delete access to the tables it uses. Other simultaneous transactions may still be able to select rows from these tables.

© 1997-2013 Devart. All Rights Reserved.

### 17.18.3 Routines

Routines in the **IBCClasses** unit.

#### Routines

Name	Description
<a href="#">Reverse2</a>	Switches places of bytes in the word argument.
<a href="#">XSQLDA_LENGTH</a>	Analogue of InterBase XSQLDA_LENGTH macro. Calculates the number of bytes that must be allocated for an input or output XSQLDA.

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.3.1 Reverse2 Function

Switches places of bytes in the word argument.

#### Unit

[IBCClasses](#)

#### Syntax

```
function Reverse2 (Value: word) : Word;
```

#### Parameters

*Value*

Holds the input value.

#### Return Value

the output value.

© 1997-2013 Devart. All Rights Reserved.

### 17.18.3.2 XSQLDA\_LENGTH Function

Analogue of InterBase XSQLDA LENGTH macro. Calculates the number of bytes that must be allocated for an input or output XSQLDA.

#### Unit

[IBCClasses](#)

#### Syntax

```
function XSQLDA_LENGTH(n: Long; XSQLVARType: TXSQLVARType): Long;
```

#### Parameters

*n*

Holds the count of XSQLVAR.

*XSQLVARType*

Holds the type of XSQLVAR.

#### Return Value

the number of bytes that must be allocated for an input or output XSQLDA.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.18.4 Variables

Variables in the **IBCClasses** unit.

#### Variables

Name	Description
<a href="#">IntegerPrecision</a>	Set this constant to define the type of NUMERIC and DECIMAL fields with precision less or equal than IntegerPrecision as dtInteger. Otherwise, they are defined as dtFloat.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.18.4.1 IntegerPrecision Variable

Set this constant to define the type of NUMERIC and DECIMAL fields with precision less or equal than IntegerPrecision as dtInteger. Otherwise, they are defined as dtFloat.

#### Unit

[IBCClasses](#)

#### Syntax

```
IntegerPrecision: integer = 10;
```

---

© 1997-2013 Devart. All Rights Reserved.

## 17.19 IBConnectionPool

This unit contains the `TIBConnectionPoolManager` class for managing connection pool.

### Classes

Name	Description
<a href="#">TIBConnectionPoolManager</a>	A class of methods that are used for managing IBDAC connection pool.

© 1997-2013 Devart. All Rights Reserved.

### 17.19.1 Classes

Classes in the **IBConnectionPool** unit.

### Classes

Name	Description
<a href="#">TIBConnectionPoolManager</a>	A class of methods that are used for managing IBDAC connection pool.

© 1997-2013 Devart. All Rights Reserved.

#### 17.19.1.1 TIBConnectionPoolManager Class

A class of methods that are used for managing IBDAC connection pool.  
For a list of all members of this type, see [TIBConnectionPoolManager](#) members.

### Unit

[IBConnectionPool](#)

### Syntax

```
TIBConnectionPoolManager = class (TCRConnectionPoolManager);
```

### Remarks

Use the `TIBConnectionPoolManager` methods to manage IBDAC connection pool.

### Inheritance Hierarchy

```
TCRConnectionPoolManager
TIBConnectionPoolManager
```

### See Also

- [Connection Pooling](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.19.1.1.1 Members

[TIBConnectionPoolManager](#) class overview.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.20 IBCErrors

IBCErrors unit implements the following classes: .

### Classes

Name	Description
<a href="#">EIBCErrors</a>	A base class for exceptions that are raised when a component detects an InterBase error.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.20.1 Classes

Classes in the **IBCErrors** unit.

### Classes

Name	Description
<a href="#">EIBCErrors</a>	A base class for exceptions that are raised when a component detects an InterBase error.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.20.1.1 EIBCErrors Class

A base class for exceptions that are raised when a component detects an InterBase error.

For a list of all members of this type, see [EIBCErrors](#) members.

### Unit

[EIBCErrors](#)

### Syntax

```
EIBCErrors = class(EIBCErrors);
```

### Remarks

EIBCErrors is raised when a component detects an InterBase error. Use EIBCErrors in an exception handling block.

### Inheritance Hierarchy

[EIBCErrors](#)

**EIBCErrors**

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.20.1.1.1 Members

[EIBCErrors](#) class overview.

### Properties

Name	Description
<a href="#">Component</a> (inherited from <a href="#">EDAErrors</a> )	Contains the component that caused the error.
<a href="#">ErrorCode</a> (inherited from <a href="#">EDAErrors</a> )	Determines the error code returned by the server.
<a href="#">ErrorNumber</a>	Used to determine the error number returned by InterBase.
<a href="#">Sender</a>	Holds the reference to the sender if exception is raised by the TComponent instance.
<a href="#">SQLErrorMsg</a>	Holds the error message describing the part of the SQL code that caused the error.

© 1997-2013 Devart. All Rights Reserved.

#### 17.20.1.1.2 Properties

Properties of the **EIBCErrors** class.

For a complete list of the **EIBCErrors** class members, see the [EIBCErrors Members](#) topic.

### Public

Name	Description
<a href="#">Component</a> (inherited from <a href="#">EDAErrors</a> )	Contains the component that caused the error.
<a href="#">ErrorCode</a> (inherited from <a href="#">EDAErrors</a> )	Determines the error code returned by the server.
<a href="#">ErrorNumber</a>	Used to determine the error number returned by InterBase.
<a href="#">Sender</a>	Holds the reference to the sender if exception is raised by the TComponent instance.
<a href="#">SQLErrorMsg</a>	Holds the error message describing the part of the SQL code that caused the error.

**See Also**

- [EIBCErrors Class](#)
  - [EIBCErrors Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.20.1.1.2.1 ErrorNumber Property

Used to determine the error number returned by InterBase.

**Class**

[EIBCErrors](#)

**Syntax**

```
property ErrorNumber: integer;
```

**Remarks**

Use the ErrorNumber property to determine the error number returned by InterBase.

**See Also**

- [EDAErrors.ErrorCode](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.20.1.1.2.2 Sender Property

Holds the reference to the sender if exception is raised by the TComponent instance.

**Class**

[EIBCErrors](#)

**Syntax**

```
property Sender: TComponent;
```

**Remarks**

The Sender property holds the reference to the sender if exception is raised by the TComponent instance.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.20.1.1.2.3 SQLErrMsg Property

Holds the error message describing the part of the SQL code that caused the error.

**Class**

[EIBCErrors](#)

**Syntax**

```
property SQLErrorMsg: string;
```

**Remarks**

The error message describing the part of the SQL code that caused the error.

© 1997-2013 Devart. All Rights Reserved.

## 17.21 IBCLoader

This unit contains implementation of the TIBCLoader component.

**Classes**

Name	Description
<a href="#">TIBCLoader</a>	This component serves for loading external data into the database table.

© 1997-2013 Devart. All Rights Reserved.

### 17.21.1 Classes

Classes in the **IBCLoader** unit.

**Classes**

Name	Description
<a href="#">TIBCLoader</a>	This component serves for loading external data into the database table.

© 1997-2013 Devart. All Rights Reserved.

#### 17.21.1.1 TIBCLoader Class

This component serves for loading external data into the database table. For a list of all members of this type, see [TIBCLoader](#) members.

**Unit**

[IBCLoader](#)

**Syntax**

```
TIBCLoader = class(TDALoader);
```

**Remarks**

The TIBCLoader component allows to load external data into the database table. TIBCLoader serves for fast loading data to the database. To specify the name of the loading table set the TableName property. Use the Columns property to access individual columns. Write the OnGetColumnData or OnPutData event handlers to read external data and pass it to the database. Call the Load method to start loading data.

TIBCLoader loads data by executing INSERT statements. For Firebird 2.0 and higher several INSERT statements are combined in one EXECUTE BLOCK statement to speed up loading.

## Inheritance Hierarchy

[TDALoader](#)  
**TIBCLoader**

---

© 1997-2013 Devart. All Rights Reserved.

17.21.1.1.1 Members

[TIBCLoader](#) class overview.

## Properties

Name	Description
<a href="#">Columns</a>	Used to access individual columns.
<a href="#">Connection</a> (inherited from <a href="#">TDALoader</a> )	Used to specify TCustomDACConnection in which TDALoader will be executed.
<a href="#">TableName</a>	Used to specify the name of the loading table set.

## Methods

Name	Description
<a href="#">CreateColumns</a> (inherited from <a href="#">TDALoader</a> )	Creates <a href="#">TDAColumn</a> objects for all fields of the table with the same name as <a href="#">TDALoader.TableName</a> .
<a href="#">Load</a> (inherited from <a href="#">TDALoader</a> )	Starts loading data.
<a href="#">LoadFromDataSet</a> (inherited from <a href="#">TDALoader</a> )	Loads data from the specified dataset.
<a href="#">PutColumnData</a> (inherited from <a href="#">TDALoader</a> )	Overloaded. Puts the value of individual columns.

## Events

Name	Description
<a href="#">OnGetColumnData</a>	Used to read external data.
<a href="#">OnProgress</a> (inherited from <a href="#">TDALoader</a> )	Occurs if handling data loading progress of the <a href="#">TDALoader.LoadFromDataSet</a> method is needed.
<a href="#">OnPutData</a>	Used to pass external data to the database.

---

© 1997-2013 Devart. All Rights Reserved.



## 17.21.1.1.2 Properties

Properties of the **TIBCLoader** class.

For a complete list of the **TIBCLoader** class members, see the [TIBCLoader Members](#) topic.

**Public**

Name	Description
<a href="#">Connection</a> (inherited from <a href="#">TDALoader</a> )	Used to specify TCustomDACConnection in which TDALoader will be executed.
<a href="#">CreateColumns</a> (inherited from <a href="#">TDALoader</a> )	Creates <a href="#">TDAColumn</a> objects for all fields of the table with the same name as <a href="#">TDALoader.TableName</a> .
<a href="#">Load</a> (inherited from <a href="#">TDALoader</a> )	Starts loading data.
<a href="#">LoadFromDataSet</a> (inherited from <a href="#">TDALoader</a> )	Loads data from the specified dataset.
<a href="#">OnGetColumnData</a> (inherited from <a href="#">TDALoader</a> )	Occurs when it is needed to put column values.
<a href="#">OnProgress</a> (inherited from <a href="#">TDALoader</a> )	Occurs if handling data loading progress of the <a href="#">TDALoader.LoadFromDataSet</a> method is needed.
<a href="#">OnPutData</a> (inherited from <a href="#">TDALoader</a> )	Occurs when putting loading data by rows is needed.
<a href="#">PutColumnData</a> (inherited from <a href="#">TDALoader</a> )	Overloaded. Puts the value of individual columns.

**Published**

Name	Description
<a href="#">Columns</a>	Used to access individual columns.
<a href="#">TableName</a>	Used to specify the name of the loading table set.

**See Also**

- [TIBCLoader Class](#)
- [TIBCLoader Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.21.1.1.2.1 Columns Property

Used to access individual columns.

**Class**

[TIBCLoader](#)

**Syntax**

**property** Columns: [TDAColumns](#);

**Remarks**

Use the Columns property to access individual columns.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.21.1.1.2.2 TableName Property

Used to specify the name of the loading table set.

**Class**

[TIBCLoader](#)

**Syntax**

**property** TableName: string;

**Remarks**

Use the TableName property to specify the name of the loading table set.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.21.1.1.3 Events

Events of the **TIBCLoader** class.

For a complete list of the **TIBCLoader** class members, see the [TIBCLoader Members](#) topic.

**Public**

Name	Description
<a href="#">Columns</a> (inherited from <a href="#">TDALoader</a> )	Used to add a <a href="#">TDAColumn</a> object for each field that will be loaded.
<a href="#">Connection</a> (inherited from <a href="#">TDALoader</a> )	Used to specify TCustomDAConnection in which TDALoader will be executed.
<a href="#">CreateColumns</a> (inherited from <a href="#">TDALoader</a> )	Creates <a href="#">TDAColumn</a> objects for all fields of the table with the same name as <a href="#">TDALoader.TableName</a> .
<a href="#">Load</a> (inherited from <a href="#">TDALoader</a> )	Starts loading data.
<a href="#">LoadFromDataSet</a> (inherited from <a href="#">TDALoader</a> )	Loads data from the specified dataset.
<a href="#">OnProgress</a> (inherited from <a href="#">TDALoader</a> )	Occurs if handling data loading progress of the <a href="#">TDALoader.LoadFromDataSet</a> method is needed.

[PutColumnData](#) (inherited from [TDALoader](#))

Overloaded. Puts the value of individual columns.

[TableName](#) (inherited from [TDALoader](#))

Used to specify the name of the table to which data will be loaded.

## Published

Name	Description
<a href="#">OnGetColumnData</a>	Used to read external data.
<a href="#">OnPutData</a>	Used to pass external data to the database.

## See Also

- [TIBCLoader Class](#)
- [TIBCLoader Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.21.1.1.3.1 OnGetColumnData Event

Used to read external data.

## Class

[TIBCLoader](#)

## Syntax

```
property OnGetColumnData: TGetColumnDataEvent;
```

## Remarks

Write the OnGetColumnData event handler to read external data.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.21.1.1.3.2 OnPutData Event

Used to pass external data to the database.

## Class

[TIBCLoader](#)

## Syntax

```
property OnPutData: TDAPutDataEvent;
```

## Remarks

Write the OnPutData event handler to pass external data to the database.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.22 IBCScript

This unit contains implementation of the TIBCScript component.

### Classes

Name	Description
<a href="#">TIBCScript</a>	A component for executing several SQL statements one by one.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.22.1 Classes

Classes in the **IBCScript** unit.

### Classes

Name	Description
<a href="#">TIBCScript</a>	A component for executing several SQL statements one by one.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.22.1.1 TIBCScript Class

A component for executing several SQL statements one by one.  
For a list of all members of this type, see [TIBCScript](#) members.

### Unit

[IBCScript](#)

### Syntax

```
TIBCScript = class (TDAScript) ;
```

### Remarks

Often it is necessary to execute several SQL statements one by one. Known way is using a lot of components such as [TIBCSQL](#). Usually it is not a good solution. With only one TIBCScript component you can execute several SQL statements as one. This sequence of statements is named script. To separate single statements use semicolon (;), slash (/), and for statements that can contain semicolon (for example CREATE TRIGGER or CREATE PROCEDURE) - only slash . Note that slash must be the first character in line.

Errors that occur while execution can be processed in the [TDAScript.OnError](#) event handler. By default, on error TIBCScript shows exception and continues execution.

### Inheritance Hierarchy

[TDAScript](#)

**TIBCScript**

## See Also

- [TIBCSQL](#)

© 1997-2013 Devart. All Rights Reserved.

17.22.1.1.1 Members

[TIBCScript](#) class overview.

## Properties

Name	Description
<a href="#">AutoDDL</a>	Used to indicate whether DDL statements must be executed in a separate transaction.
<a href="#">Connection</a>	Used to specify the connection in which the script will be executed.
<a href="#">DataSet</a>	Used to get or set script result dataset.
<a href="#">Debug</a> (inherited from <a href="#">TDAScript</a> )	Used to display the script execution and all its parameter values.
<a href="#">Delimiter</a> (inherited from <a href="#">TDAScript</a> )	Used to set the delimiter string that separates script statements.
<a href="#">EndLine</a> (inherited from <a href="#">TDAScript</a> )	Used to get the current statement last line number in a script.
<a href="#">EndOffset</a> (inherited from <a href="#">TDAScript</a> )	Used to get the offset in the last line of the current statement.
<a href="#">EndPos</a> (inherited from <a href="#">TDAScript</a> )	Used to get the end position of the current statement.
<a href="#">Macros</a> (inherited from <a href="#">TDAScript</a> )	Used to change SQL script text in design- or run-time easily.
<a href="#">Params</a>	Used to hold the parameters for a SQL script.
<a href="#">SQL</a> (inherited from <a href="#">TDAScript</a> )	Used to get or set script text.
<a href="#">StartLine</a> (inherited from <a href="#">TDAScript</a> )	Used to get the current statement start line number in a script.
<a href="#">StartOffset</a> (inherited from <a href="#">TDAScript</a> )	Used to get the offset in the first line of the current statement.

[StartPos](#) (inherited from [TDA Script](#))

Used to get the start position of the current statement in a script.

[Statements](#) (inherited from [TDA Script](#))

Contains a list of statements obtained from the SQL property.

[Transaction](#)

Used to set or return the transaction to be used by the component.

## Methods

Name	Description
<a href="#">BreakExec</a> (inherited from <a href="#">TDA Script</a> )	Stops script execution.
<a href="#">ErrorOffset</a> (inherited from <a href="#">TDA Script</a> )	Used to get the offset of the statement if the Execute method raised an exception.
<a href="#">Execute</a> (inherited from <a href="#">TDA Script</a> )	Executes a script.
<a href="#">ExecuteFile</a> (inherited from <a href="#">TDA Script</a> )	Executes SQL statements contained in a file.
<a href="#">ExecuteNext</a> (inherited from <a href="#">TDA Script</a> )	Executes the next statement in the script and then stops.
<a href="#">ExecuteStream</a> (inherited from <a href="#">TDA Script</a> )	Executes SQL statements contained in a stream object.
<a href="#">FindMacro</a> (inherited from <a href="#">TDA Script</a> )	Indicates whether a specified macro exists in a dataset.
<a href="#">MacroByName</a> (inherited from <a href="#">TDA Script</a> )	Finds a Macro with the name passed in Name.

## Events

Name	Description
<a href="#">AfterExecute</a> (inherited from <a href="#">TDA Script</a> )	Occurs after a SQL script execution.
<a href="#">BeforeExecute</a> (inherited from <a href="#">TDA Script</a> )	Occurs when taking a specific action before executing the current SQL statement is needed.
<a href="#">OnError</a> (inherited from <a href="#">TDA Script</a> )	Occurs when InterBase raises an error.

© 1997-2013 Devart. All Rights Reserved.

### 17.22.1.1.2 Properties

Properties of the **TIBCScript** class.  
For a complete list of the **TIBCScript** class members, see the [TIBCScript Members](#) topic.

**Public**

Name	Description
<a href="#">BreakExec</a> (inherited from <a href="#">TDAScript</a> )	Stops script execution.
<a href="#">EndLine</a> (inherited from <a href="#">TDAScript</a> )	Used to get the current statement last line number in a script.
<a href="#">EndOffset</a> (inherited from <a href="#">TDAScript</a> )	Used to get the offset in the last line of the current statement.
<a href="#">EndPos</a> (inherited from <a href="#">TDAScript</a> )	Used to get the end position of the current statement.
<a href="#">ErrorOffset</a> (inherited from <a href="#">TDAScript</a> )	Used to get the offset of the statement if the Execute method raised an exception.
<a href="#">Execute</a> (inherited from <a href="#">TDAScript</a> )	Executes a script.
<a href="#">ExecuteFile</a> (inherited from <a href="#">TDAScript</a> )	Executes SQL statements contained in a file.
<a href="#">ExecuteNext</a> (inherited from <a href="#">TDAScript</a> )	Executes the next statement in the script and then stops.
<a href="#">ExecuteStream</a> (inherited from <a href="#">TDAScript</a> )	Executes SQL statements contained in a stream object.
<a href="#">FindMacro</a> (inherited from <a href="#">TDAScript</a> )	Indicates whether a specified macro exists in a dataset.
<a href="#">MacroByName</a> (inherited from <a href="#">TDAScript</a> )	Finds a Macro with the name passed in Name.
<a href="#">Params</a>	Used to hold the parameters for a SQL script.
<a href="#">StartLine</a> (inherited from <a href="#">TDAScript</a> )	Used to get the current statement start line number in a script.
<a href="#">StartOffset</a> (inherited from <a href="#">TDAScript</a> )	Used to get the offset in the first line of the current statement.
<a href="#">StartPos</a> (inherited from <a href="#">TDAScript</a> )	Used to get the start position of the current statement in a script.
<a href="#">Statements</a> (inherited from <a href="#">TDAScript</a> )	Contains a list of statements obtained from the SQL property.

**Published**

Name	Description
<a href="#">AfterExecute</a> (inherited from <a href="#">TDAScript</a> )	Occurs after a SQL script execution.

[AutoDDL](#)

Used to indicate whether DDL statements must be executed in a separate transaction.

[BeforeExecute](#) (inherited from [TDAScript](#))

Occurs when taking a specific action before executing the current SQL statement is needed.

[Connection](#)

Used to specify the connection in which the script will be executed.

[DataSet](#)

Used to get or set script result dataset.

[Debug](#) (inherited from [TDAScript](#))

Used to display the script execution and all its parameter values.

[Delimiter](#) (inherited from [TDAScript](#))

Used to set the delimiter string that separates script statements.

[Macros](#) (inherited from [TDAScript](#))

Used to change SQL script text in design- or run-time easily.

[OnError](#) (inherited from [TDAScript](#))

Occurs when InterBase raises an error.

[SQL](#) (inherited from [TDAScript](#))

Used to get or set script text.

[Transaction](#)

Used to set or return the transaction to be used by the component.

**See Also**

- [TIBCScript Class](#)
  - [TIBCScript Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.22.1.1.2.1 AutoDDL Property

Used to indicate whether DDL statements must be executed in a separate transaction.

**Class**[TIBCScript](#)**Syntax**

```
property AutoDDL: Boolean default True;
```

**Remarks**

Use the AutoDDL property to determine whether DDL statements must be executed in a separate transaction.

---



© 1997-2013 Devart. All Rights Reserved.

#### 17.22.1.1.2.2 Connection Property

Used to specify the connection in which the script will be executed.

### Class

[TIBCScript](#)

### Syntax

**property** Connection: [TIBCCConnection](#);

### Remarks

Use the Connection property to specify the connection in which the script will be executed. If connection is not connected Execute method calls Connection.Connect.

### See Also

- [TIBCCConnection](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.22.1.1.2.3 DataSet Property

Used to get or set script result dataset.

### Class

[TIBCScript](#)

### Syntax

**property** DataSet: [TCustomIBCDataset](#);

### Remarks

Use the DataSet property to get or set script result dataset.

© 1997-2013 Devart. All Rights Reserved.

#### 17.22.1.1.2.4 Params Property

Used to hold the parameters for a SQL script.

### Class

[TIBCScript](#)

### Syntax

**property** Params: [TIBCPParams](#);

### Remarks

Contains the parameters for a SQL script.  
Access Params at runtime to view and set parameter names, values, and data types

dynamically (at design time use the Parameters editor to set parameter information). Params is a zero-based array of parameter records. Index specifies the array element to access.

An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName.

### See Also

- [TIBCTParam](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.22.1.1.2.5 Transaction Property

Used to set or return the transaction to be used by the component.

### Class

[TIBCTScript](#)

### Syntax

**property** Transaction: [TIBCTTransaction](#) **stored** IsTransactionStored;

### Remarks

Use the Transaction property to set or return the transaction to be used by the component.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.23 IBCSQLMonitor

This unit contains implementation of the TIBCSQLMonitor component.

### Classes

Name	Description
<a href="#">TIBCSQLMonitor</a>	[Inherited]

---

© 1997-2013 Devart. All Rights Reserved.

### 17.23.1 Classes

Classes in the **IBCSQLMonitor** unit.

### Classes

Name	Description
<a href="#">TIBCSQLMonitor</a>	[Inherited]

---

© 1997-2013 Devart. All Rights Reserved.

### 17.23.1.1 TIBCSQLMonitor Class

[Inherited

For a list of all members of this type, see [TIBCSQLMonitor](#) members.

#### Unit

[TIBCSQLMonitor](#)

#### Syntax

```
TIBCSQLMonitor = class(TCustomDASQLMonitor);
```

#### Remarks

Use TIBCSQLMonitor to monitor dynamic SQL execution in IBDAC-based applications. TIBCSQLMonitor provides two ways of displaying debug information: with dialog window, A:Using DBMonitor or Borland SQL Monitor. Furthermore to receive debug information the [TCustomDASQLMonitor.OnSQL](#) event can be used. Also it is possible to use all these ways at the same time, though an application may have only one TIBCSQLMonitor object. If an application has no TIBCSQLMonitor instance, the Debug window is available to display SQL statements to be sent.

#### Inheritance Hierarchy

[TCustomDASQLMonitor](#)  
**TIBCSQLMonitor**

#### See Also

- [TCustomDADataset.Debug](#)
- [TCustomDASQL.Debug](#)
- A:Using DBMonitor

© 1997-2013 Devart. All Rights Reserved.

#### 17.23.1.1.1 Members

[TIBCSQLMonitor](#) class overview.

#### Properties

Name	Description
<a href="#">Active</a> (inherited from <a href="#">TCustomDASQLMonitor</a> )	Used to activate monitoring of SQL.
<a href="#">DBMonitorOptions</a> (inherited from <a href="#">TCustomDASQLMonitor</a> )	Used to set options for dbMonitor.
<a href="#">Options</a> (inherited from <a href="#">TCustomDASQLMonitor</a> )	Used to include the desired properties for TCustomDASQLMonitor.
<a href="#">TraceFlags</a> (inherited from <a href="#">TCustomDASQLMonitor</a> )	Used to specify which database operations the monitor should track in an application at runtime.

**Events**

Name	Description
<a href="#">OnSQL</a> (inherited from <a href="#">TCustomDASQLMonitor</a> )	Occurs when tracing of SQL activity on database components is needed.

---

© 1997-2013 Devart. All Rights Reserved.

**17.24 IbDacVcl**

This unit contains the visual constituent of IBDAC.

**Classes**

Name	Description
<a href="#">TIBConnectDialog</a>	A class that provides a dialog box for user to supply his login information.

---

© 1997-2013 Devart. All Rights Reserved.

**17.24.1 Classes**

Classes in the **IbDacVcl** unit.

**Classes**

Name	Description
<a href="#">TIBConnectDialog</a>	A class that provides a dialog box for user to supply his login information.

---

© 1997-2013 Devart. All Rights Reserved.

**17.24.1.1 TIBConnectDialog Class**

A class that provides a dialog box for user to supply his login information.  
For a list of all members of this type, see [TIBConnectDialog](#) members.

**Unit**

[IbDacVcl](#)

**Syntax**

```
TIBConnectDialog = class (TCustomConnectDialog) ;
```

**Remarks**

The TIBConnectDialog component is a direct descendant of TCustomConnectDialog class. Use TIBConnectDialog to provide dialog box for user to supply server name, user name, and password. You may want to customize appearance of dialog box using this class's properties.

## Inheritance Hierarchy

[TCustomConnectDialog](#)

**TIBCCConnectDialog**

## See Also

- [TCustomDAConnection.ConnectDialog](#)

© 1997-2013 Devart. All Rights Reserved.

17.24.1.1.1 Members

[TIBCCConnectDialog](#) class overview.

## Properties

Name	Description
<a href="#">CancelButton</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to specify the label for the Cancel button.
<a href="#">Caption</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to set the caption of dialog box.
<a href="#">ConnectButton</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to specify the label for the Connect button.
<a href="#">Connection</a>	Used to indicate the TIBCCConnection component using the TIBCCConnectDialog object.
<a href="#">DatabaseLabel</a>	Used to specify a prompt for database edit.
<a href="#">DialogClass</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to specify the class of the form that will be displayed to enter login information.
<a href="#">LabelSet</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to set the language of buttons and labels captions.
<a href="#">PasswordLabel</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to specify a prompt for password edit.
<a href="#">ProtocolLabel</a>	Used to specify a prompt for protocol box.
<a href="#">Retries</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to indicate the number of retries of failed connections.
<a href="#">SavePassword</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used for the password to be displayed in ConnectDialog in asterisks.
<a href="#">ServerLabel</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to specify a prompt for the server name edit.

[StoreLogInfo](#) (inherited from [TCustomConnectDialog](#))

Used to specify whether the login information should be kept in system registry after a connection was established.

[UsernameLabel](#) (inherited from [TCustomConnectDialog](#))

Used to specify a prompt for username edit.

## Methods

Name	Description
<a href="#">Execute</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button.
<a href="#">GetServerList</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Retrieves a list of available server names.

© 1997-2013 Devart. All Rights Reserved.

### 17.24.1.1.2 Properties

Properties of the **TIBConnectDialog** class.

For a complete list of the **TIBConnectDialog** class members, see the [TIBConnectDialog Members](#) topic.

## Public

Name	Description
<a href="#">CancelButton</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to specify the label for the Cancel button.
<a href="#">Caption</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to set the caption of dialog box.
<a href="#">ConnectButton</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to specify the label for the Connect button.
<a href="#">Connection</a>	Used to indicate the TIBConnection component using the TIBConnectDialog object.
<a href="#">DialogClass</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to specify the class of the form that will be displayed to enter login information.
<a href="#">Execute</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button.
<a href="#">GetServerList</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Retrieves a list of available server names.
<a href="#">LabelSet</a> (inherited from <a href="#">TCustomConnectDialog</a> )	Used to set the language of buttons and labels captions.

[PasswordLabel](#) (inherited from [TCustomConnectDialog](#))

Used to specify a prompt for password edit.

[Retries](#) (inherited from [TCustomConnectDialog](#))

Used to indicate the number of retries of failed connections.

[SavePassword](#) (inherited from [TCustomConnectDialog](#))

Used for the password to be displayed in ConnectDialog in asterisks.

[ServerLabel](#) (inherited from [TCustomConnectDialog](#))

Used to specify a prompt for the server name edit.

[StoreLogInfo](#) (inherited from [TCustomConnectDialog](#))

Used to specify whether the login information should be kept in system registry after a connection was established.

[UsernameLabel](#) (inherited from [TCustomConnectDialog](#))

Used to specify a prompt for username edit.

## Published

Name	Description
<a href="#">DatabaseLabel</a>	Used to specify a prompt for database edit.
<a href="#">ProtocolLabel</a>	Used to specify a prompt for protocol box.

## See Also

- [TIBCCConnectDialog Class](#)
- [TIBCCConnectDialog Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.24.1.1.2.1 Connection Property

Used to indicate the TIBCCConnection component using the TIBCCConnectDialog object.

## Class

[TIBCCConnectDialog](#)

## Syntax

**property** Connection: [TIBCCConnection](#);

## Remarks

Read the Connection property to learn what TIBCCConnection component uses the TIBCCConnectDialog object. This property is read only.

## See Also

- [TCustomDAConnection.ConnectDialog](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.24.1.1.2.2 DatabaseLabel Property

Used to specify a prompt for database edit.

#### Class

[TIBConnectDialog](#)

#### Syntax

```
property DatabaseLabel: string;
```

#### Remarks

Use the DatabaseLabel property to specify a prompt for database edit.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.24.1.1.2.3 ProtocolLabel Property

Used to specify a prompt for protocol box.

#### Class

[TIBConnectDialog](#)

#### Syntax

```
property ProtocolLabel: string;
```

#### Remarks

Use the ProtocolLabel property to specify a prompt for protocol box.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.25 MemData

This unit contains classes for storing data in memory.

#### Classes

Name	Description
<a href="#">TAttribute</a>	TAttribute is not used in IBDAC.
<a href="#">TBlob</a>	Holds large object value for field and parameter dtBlob, dtMemo data types.
<a href="#">TCompressedBlob</a>	Holds large object value for field and parameter dtBlob, dtMemo data types and can compress its data.
<a href="#">TDBObject</a>	A base class for classes that work with user-defined data types that have attributes.



[TObjectType](#)  
[TSharedObject](#)

This class is not used.  
 A base class that allows to simplify memory management for object referenced by several other objects.

## Types

Name	Description
<a href="#">TLocateExOptions</a>	Represents the set of <a href="#">TLocateExOption</a> .
<a href="#">TUpdateReckinds</a>	Represents the set of TUpdateReckind.

## Enumerations

Name	Description
<a href="#">TCompressBlobMode</a>	Specifies when the values should be compressed and the way they should be stored.
<a href="#">TConnLostCause</a>	Specifies the cause of the connection loss.
<a href="#">TDANumericType</a>	Specifies the format of storing and representing of the NUMERIC (DECIMAL) fields.
<a href="#">TLocateExOption</a>	Allows to set additional search parameters which will be used by the LocateEx method.
<a href="#">TSortType</a>	Specifies a sort type for string fields.
<a href="#">TUpdateReckind</a>	Indicates records for which the ApplyUpdates method will be performed.

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1 Classes

Classes in the **MemData** unit.

## Classes

Name	Description
<a href="#">TAttribute</a>	TAttribute is not used in IBDAC.
<a href="#">TBlob</a>	Holds large object value for field and parameter dtBlob, dtMemo data types.

[TCompressedBlob](#)

Holds large object value for field and parameter dtBlob, dtMemo data types and can compress its data.

[TDBObject](#)

A base class for classes that work with user-defined data types that have attributes.

[TObjectType](#)

This class is not used.

[TSharedObject](#)

A base class that allows to simplify memory management for object referenced by several other objects.

© 1997-2013 Devart. All Rights Reserved.

### 17.25.1.1 TAttribute Class

TAttribute is not used in IBDAC.

For a list of all members of this type, see [TAttribute](#) members.

#### Unit

[MemData](#)

#### Syntax

```
TAttribute = class (System.TObject);
```

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.1.1 Members

[TAttribute](#) class overview.

#### Properties

Name	Description
<a href="#">AttributeNo</a>	Returns an attribute's ordinal position in object.
<a href="#">DataSize</a>	Returns the size of an attribute value in internal representation.
<a href="#">DataType</a>	Returns the type of data that was assigned to the Attribute.
<a href="#">Length</a>	Returns the length of the string for dtString attribute and precision for dtInteger and dtFloat attribute.
<a href="#">ObjectType</a>	Returns a TObjectType object for an object attribute.

<a href="#">Offset</a>	Returns an offset of the attribute value in internal representation.
<a href="#">Owner</a>	Indicates TObjectType that uses the attribute to represent one of its attributes.
<a href="#">Scale</a>	Returns the scale of dtFloat and dtInteger attributes.
<a href="#">Size</a>	Returns the size of an attribute value in external representation.

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.1.2 Properties

Properties of the **TAttribute** class.

For a complete list of the **TAttribute** class members, see the [TAttribute Members](#) topic.

#### Public

Name	Description
<a href="#">AttributeNo</a>	Returns an attribute's ordinal position in object.
<a href="#">DataSize</a>	Returns the size of an attribute value in internal representation.
<a href="#">DataType</a>	Returns the type of data that was assigned to the Attribute.
<a href="#">Length</a>	Returns the length of the string for dtString attribute and precision for dtInteger and dtFloat attribute.
<a href="#">ObjectType</a>	Returns a TObjectType object for an object attribute.
<a href="#">Offset</a>	Returns an offset of the attribute value in internal representation.
<a href="#">Owner</a>	Indicates TObjectType that uses the attribute to represent one of its attributes.
<a href="#">Scale</a>	Returns the scale of dtFloat and dtInteger attributes.
<a href="#">Size</a>	Returns the size of an attribute value in external representation.

**See Also**

- [TAttribute Class](#)
  - [TAttribute Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.1.2.1 AttributeNo Property

Returns an attribute's ordinal position in object.

**Class**

[TAttribute](#)

**Syntax**

```
property AttributeNo: Word;
```

**Remarks**

Use the AttributeNo property to learn an attribute's ordinal position in object, where 1 is the first field.

**See Also**

- [TObjectType.Attributes](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.1.2.2 DataSize Property

Returns the size of an attribute value in internal representation.

**Class**

[TAttribute](#)

**Syntax**

```
property DataSize: Integer;
```

**Remarks**

Use the DataSize property to learn the size of an attribute value in internal representation.

For example:

dtDate	17 (sizeof (OCIDate))
dtFloat	22 (sizeof (OCINumber))
dtInteger	22 (sizeof (OCINumber))

---

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.1.2.3 DataType Property

Returns the type of data that was assigned to the Attribute.

**Class**

[TAttribute](#)

**Syntax**

```
property DataType: Word;
```

**Remarks**

Use the DataType property to discover the type of data that was assigned to the Attribute.

Possible values: dtDate, dtFloat, dtInteger, dtString, dtObject.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.1.2.4 Length Property

Returns the length of the string for dtString attribute and precision for dtInteger and dtFloat attribute.

**Class**

[TAttribute](#)

**Syntax**

```
property Length: Word;
```

**Remarks**

Use the Length property to learn the length of the string for dtString attribute and precision for dtInteger and dtFloat attribute.

**See Also**

- [Scale](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.1.2.5 ObjectType Property

Returns a TObjectType object for an object attribute.

**Class**

[TAttribute](#)

**Syntax**

```
property ObjectType: TObjectType;
```

**Remarks**

Use the ObjectType property to return a TObjectType object for an object attribute.

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.1.2.6 Offset Property

Returns an offset of the attribute value in internal representation.

##### **Class**

[TAttribute](#)

##### **Syntax**

```
property Offset: Integer;
```

##### **Remarks**

Use the DataSize property to learn an offset of the attribute value in internal representation.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.1.2.7 Owner Property

Indicates TObjectType that uses the attribute to represent one of its attributes.

##### **Class**

[TAttribute](#)

##### **Syntax**

```
property Owner: TObjectType;
```

##### **Remarks**

Check the value of the Owner property to determine TObjectType that uses the attribute to represent one of its attributes. Applications should not assign the Owner property directly.  
It is assigned automatically when the attribute is created from a TOraType.Describe.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.1.2.8 Scale Property

Returns the scale of dtFloat and dtInteger attributes.

##### **Class**

[TAttribute](#)

##### **Syntax**

```
property Scale: Word;
```

##### **Remarks**

Use the Scale property to learn the scale of dtFloat and dtInteger attributes.

##### **See Also**

- [Length](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.1.2.9 Size Property

Returns the size of an attribute value in external representation.

### Class

[TAttribute](#)

### Syntax

```
property Size: Integer;
```

### Remarks

Read Size to learn the size of an attribute value in external representation.  
For example:

dtDate	8 (sizeof (TDateTime))
dtFloat	8 (sizeof (Double))
dtInteger	4 (sizeof (Integer))

### See Also

- [DataSize](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.2 TBlob Class

Holds large object value for field and parameter dtBlob, dtMemo data types.  
For a list of all members of this type, see [TBlob](#) members.

### Unit

[MemData](#)

### Syntax

```
TBlob = class(TSharedObject);
```

### Remarks

Object TBlob holds large object value for the field and parameter dtBlob, dtMemo, dtWideMemo data types.

### Inheritance Hierarchy

[TSharedObject](#)  
**TBlob**

## See Also

- [TMemDataSet.GetBlob](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.25.1.2.1 Members

[TBlob](#) class overview.

## Properties

Name	Description
<a href="#">AsString</a>	Used to manipulate BLOB value as string.
<a href="#">AsWideString</a>	Used to manipulate BLOB value as Unicode string.
<a href="#">IsUnicode</a>	Gives choice of making TBlob store and process data in Unicode format or not.
<a href="#">RefCount</a> (inherited from <a href="#">TSharedObject</a> )	Used to return the count of reference to a TSharedObject object.
<a href="#">Size</a>	Used to learn the size of the TBlob value in bytes.

## Methods

Name	Description
<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">Assign</a>	Sets BLOB value from another TBlob object.
<a href="#">Clear</a>	Deletes the current value in TBlob object.
<a href="#">LoadFromFile</a>	Loads the contents of a file into a TBlob object.
<a href="#">LoadFromStream</a>	Copies the contents of a stream into the TBlob object.
<a href="#">Read</a>	Acquires a raw sequence of bytes from the data stored in TBlob.
<a href="#">Release</a> (inherited from <a href="#">TSharedObject</a> )	Decrements the reference count.
<a href="#">SaveToFile</a>	Saves the contents of the TBlob object to a file.



[SaveToStream](#)

[Truncate](#)

[Write](#)

Copies the contents of a TBlob object to a stream.

Sets new TBlob size and discards all data over it.

Stores a raw sequence of bytes into a TBlob object.

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.2.2 Properties

Properties of the **TBlob** class.

For a complete list of the **TBlob** class members, see the [TBlob Members](#) topic.

#### Public

Name	Description
<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">AsString</a>	Used to manipulate BLOB value as string.
<a href="#">AsWideString</a>	Used to manipulate BLOB value as Unicode string.
<a href="#">IsUnicode</a>	Gives choice of making TBlob store and process data in Unicode format or not.
<a href="#">RefCount</a> (inherited from <a href="#">TSharedObject</a> )	Used to return the count of reference to a TSharedObject object.
<a href="#">Release</a> (inherited from <a href="#">TSharedObject</a> )	Decrements the reference count.
<a href="#">Size</a>	Used to learn the size of the TBlob value in bytes.

#### See Also

- [TBlob Class](#)
- [TBlob Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.2.2.1 AsString Property

Used to manipulate BLOB value as string.

#### Class

[TBlob](#)

#### Syntax

```
property AsString: string;
```

**Remarks**

Use the AsString property to manipulate BLOB value as string.

**See Also**

- [Assign](#)
  - [AsWideString](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.2.2.2 AsWideString Property

Used to manipulate BLOB value as Unicode string.

**Class**

[TBlob](#)

**Syntax**

```
property AsWideString: string;
```

**Remarks**

Use the AsWideString property to manipulate BLOB value as Unicode string.

**See Also**

- [Assign](#)
  - [AsString](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.2.2.3 IsUnicode Property

Gives choice of making TBlob store and process data in Unicode format or not.

**Class**

[TBlob](#)

**Syntax**

```
property IsUnicode: boolean;
```

**Remarks**

Set IsUnicode to True if you want TBlob to store and process data in Unicode format.

**Note:** changing this property raises an exception if TBlob is not empty.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.2.2.4 Size Property

Used to learn the size of the TBlob value in bytes.

**Class**

[TBlob](#)

**Syntax**

```
property Size: Cardinal;
```

**Remarks**

Use the Size property to find out the size of the TBlob value in bytes.

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.2.3 Methods

Methods of the **TBlob** class.

For a complete list of the **TBlob** class members, see the [TBlob Members](#) topic.

**Public**

Name	Description
<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">Assign</a>	Sets BLOB value from another TBlob object.
<a href="#">Clear</a>	Deletes the current value in TBlob object.
<a href="#">LoadFromFile</a>	Loads the contents of a file into a TBlob object.
<a href="#">LoadFromStream</a>	Copies the contents of a stream into the TBlob object.
<a href="#">Read</a>	Acquires a raw sequence of bytes from the data stored in TBlob.
<a href="#">RefCount</a> (inherited from <a href="#">TSharedObject</a> )	Used to return the count of reference to a TSharedObject object.
<a href="#">Release</a> (inherited from <a href="#">TSharedObject</a> )	Decrements the reference count.
<a href="#">SaveToFile</a>	Saves the contents of the TBlob object to a file.
<a href="#">SaveToStream</a>	Copies the contents of a TBlob object to a stream.
<a href="#">Truncate</a>	Sets new TBlob size and discards all data over it.

[Write](#)

Stores a raw sequence of bytes into a TBlob object.

**See Also**

- [TBlob Class](#)
  - [TBlob Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.2.3.1 Assign Method

Sets BLOB value from another TBlob object.

**Class**

[TBlob](#)

**Syntax**

```
procedure Assign(Source: TBlob);
```

**Parameters***Source*

Holds the BLOB from which the value to the current object will be assigned.

**Remarks**

Call the Assign method to set BLOB value from another TBlob object.

**See Also**

- [LoadFromStream](#)
  - [AsString](#)
  - [AsWideString](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.2.3.2 Clear Method

Deletes the current value in TBlob object.

**Class**

[TBlob](#)

**Syntax**

```
procedure Clear; virtual;
```

**Remarks**

Call the Clear method to delete the current value in TBlob object.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.2.3.3 LoadFromFile Method

Loads the contents of a file into a TBlob object.

**Class**

[TBlob](#)

**Syntax**

```
procedure LoadFromFile(const FileName: string);
```

**Parameters**

*FileName*

Holds the name of the file from which the TBlob value is loaded.

**Remarks**

Call the LoadFromFile method to load the contents of a file into a TBlob object. Specify the name of the file to load into the field as the value of the FileName parameter.

**See Also**

- [SaveToFile](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.2.3.4 LoadFromStream Method

Copies the contents of a stream into the TBlob object.

**Class**

[TBlob](#)

**Syntax**

```
procedure LoadFromStream(Stream: TStream); virtual;
```

**Parameters**

*Stream*

Holds the specified stream from which the field's value is copied.

**Remarks**

Call the LoadFromStream method to copy the contents of a stream into the TBlob object. Specify the stream from which the field's value is copied as the value of the Stream parameter.

**See Also**

- [SaveToStream](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.2.3.5 Read Method

Acquires a raw sequence of bytes from the data stored in TBlob.

**Class**

[TBlob](#)

**Syntax**

```
function Read(Position: Cardinal; Count: Cardinal; Dest: IntPtr):  
Cardinal; virtual;
```

**Parameters***Position*

Holds the starting point of the byte sequence.

*Count*

Holds the size of the sequence in bytes.

*Dest*

Holds a pointer to the memory area where to store the sequence.

**Return Value**

Actually read byte count if the sequence crosses object size limit.

**Remarks**

Call the Read method to acquire a raw sequence of bytes from the data stored in TBlob.

The Position parameter is the starting point of byte sequence which lasts Count number of bytes. The Dest parameter is a pointer to the memory area where to store the sequence.

If the sequence crosses object size limit, function will return actually read byte count.

**See Also**

- [Write](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.2.3.6 SaveToFile Method

Saves the contents of the TBlob object to a file.

**Class**

[TBlob](#)

**Syntax**

```
procedure SaveToFile(const FileName: string);
```

**Parameters***FileName*

Holds a string that contains the name of the file.

## Remarks

Call the SaveToFile method to save the contents of the TBlob object to a file. Specify the name of the file as the value of the FileName parameter.

## See Also

- [LoadFromFile](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.25.1.2.3.7 SaveToStream Method

Copies the contents of a TBlob object to a stream.

## Class

[TBlob](#)

## Syntax

```
procedure SaveToStream(Stream: TStream); virtual;
```

### Parameters

*Stream*

Holds the name of the stream.

## Remarks

Call the SaveToStream method to copy the contents of a TBlob object to a stream. Specify the name of the stream to which the field's value is saved as the value of the Stream parameter.

## See Also

- [LoadFromStream](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.25.1.2.3.8 Truncate Method

Sets new TBlob size and discards all data over it.

## Class

[TBlob](#)

## Syntax

```
procedure Truncate(NewSize: Cardinal); virtual;
```

### Parameters

*NewSize*

Holds the new size of TBlob.

**Remarks**

Call the Truncate method to set new TBlob size and discard all data over it. If NewSize is greater or equal TBlob.Size, it does nothing.

---

© 1997-2013 Devart. All Rights Reserved.

**17.25.1.2.3.9 Write Method**

Stores a raw sequence of bytes into a TBlob object.

**Class**

[TBlob](#)

**Syntax**

```
procedure Write(Position: Cardinal; Count: Cardinal; Source:
  IntPtr); virtual;
```

**Parameters***Position*

Holds the starting point of the byte sequence.

*Count*

Holds the size of the sequence in bytes.

*Source*

Holds a pointer to a source memory area.

**Remarks**

Call the Write method to store a raw sequence of bytes into a TBlob object. The Position parameter is the starting point of byte sequence which lasts Count number of bytes. The Source parameter is a pointer to a source memory area. If the value of the Position parameter crosses current size limit of TBlob object, source data will be appended to the object data.

**See Also**

- [Read](#)
- 

© 1997-2013 Devart. All Rights Reserved.

**17.25.1.3 TCompressedBlob Class**

Holds large object value for field and parameter dtBlob, dtMemo data types and can compress its data.

For a list of all members of this type, see [TCompressedBlob](#) members.

**Unit**

[MemData](#)

**Syntax**

```
TCompressedBlob = class (TBlob);
```



## Remarks

TCompressedBlob is a descendant of the TBlob class. It holds large object value for field and parameter dtBlob, dtMemo data types and can compress its data. For more information about using BLOB compression see [TCustomDADataset.Options](#).

**Note:** Internal compression functions are available in CodeGear Delphi 2007 for Win32, Borland Developer Studio 2006, Borland Delphi 2005, and Borland Delphi 7. To use BLOB compression under Borland Delphi 6, Borland Delphi 5 and Borland C++ Builder you should use your own compression functions. To use them set the CompressProc and UncompressProc variables declared in the MemUtils unit.

## Example

```
type
  TCompressProc = function(dest: IntPtr; destLen: IntPtr; const source: IntPtr;
  TUncompressProc = function(dest: IntPtr; destLen: IntPtr; source: IntPtr; sou
var
  CompressProc: TCompressProc;
  UncompressProc: TUncompressProc;
```

## Inheritance Hierarchy

[TSharedObject](#)

[TBlob](#)

**TCompressedBlob**

## See Also

- [TBlob](#)
- [TMemDataSet.GetBlob](#)
- [TCustomDADataset.Options](#)

© 1997-2013 Devart. All Rights Reserved.

17.25.1.3.1 Members

[TCompressedBlob](#) class overview.

## Properties

Name	Description
<a href="#">AsString</a> (inherited from <a href="#">TBlob</a> )	Used to manipulate BLOB value as string.
<a href="#">AsWideString</a> (inherited from <a href="#">TBlob</a> )	Used to manipulate BLOB value as Unicode string.
<a href="#">Compressed</a>	Used to indicate if the Blob is compressed.
<a href="#">CompressedSize</a>	Used to indicate compressed size of the Blob data.

[IsUnicode](#) (inherited from [TBlob](#))

Gives choice of making TBlob store and process data in Unicode format or not.

[RefCount](#) (inherited from [TSharedObject](#))

Used to return the count of reference to a TSharedObject object.

[Size](#) (inherited from [TBlob](#))

Used to learn the size of the TBlob value in bytes.

## Methods

Name	Description
<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">Assign</a> (inherited from <a href="#">TBlob</a> )	Sets BLOB value from another TBlob object.
<a href="#">Clear</a> (inherited from <a href="#">TBlob</a> )	Deletes the current value in TBlob object.
<a href="#">LoadFromFile</a> (inherited from <a href="#">TBlob</a> )	Loads the contents of a file into a TBlob object.
<a href="#">LoadFromStream</a> (inherited from <a href="#">TBlob</a> )	Copies the contents of a stream into the TBlob object.
<a href="#">Read</a> (inherited from <a href="#">TBlob</a> )	Acquires a raw sequence of bytes from the data stored in TBlob.
<a href="#">Release</a> (inherited from <a href="#">TSharedObject</a> )	Decrements the reference count.
<a href="#">SaveToFile</a> (inherited from <a href="#">TBlob</a> )	Saves the contents of the TBlob object to a file.
<a href="#">SaveToStream</a> (inherited from <a href="#">TBlob</a> )	Copies the contents of a TBlob object to a stream.
<a href="#">Truncate</a> (inherited from <a href="#">TBlob</a> )	Sets new TBlob size and discards all data over it.
<a href="#">Write</a> (inherited from <a href="#">TBlob</a> )	Stores a raw sequence of bytes into a TBlob object.

© 1997-2013 Devart. All Rights Reserved.

### 17.25.1.3.2 Properties

Properties of the **TCompressedBlob** class.

For a complete list of the **TCompressedBlob** class members, see the [TCompressedBlob Members](#) topic.

## Public

Name	Description
------	-------------

<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">Assign</a> (inherited from <a href="#">TBlob</a> )	Sets BLOB value from another TBlob object.
<a href="#">AsString</a> (inherited from <a href="#">TBlob</a> )	Used to manipulate BLOB value as string.
<a href="#">AsWideString</a> (inherited from <a href="#">TBlob</a> )	Used to manipulate BLOB value as Unicode string.
<a href="#">Clear</a> (inherited from <a href="#">TBlob</a> )	Deletes the current value in TBlob object.
<a href="#">Compressed</a>	Used to indicate if the Blob is compressed.
<a href="#">CompressedSize</a>	Used to indicate compressed size of the Blob data.
<a href="#">IsUnicode</a> (inherited from <a href="#">TBlob</a> )	Gives choice of making TBlob store and process data in Unicode format or not.
<a href="#">LoadFromFile</a> (inherited from <a href="#">TBlob</a> )	Loads the contents of a file into a TBlob object.
<a href="#">LoadFromStream</a> (inherited from <a href="#">TBlob</a> )	Copies the contents of a stream into the TBlob object.
<a href="#">Read</a> (inherited from <a href="#">TBlob</a> )	Acquires a raw sequence of bytes from the data stored in TBlob.
<a href="#">RefCount</a> (inherited from <a href="#">TSharedObject</a> )	Used to return the count of reference to a TSharedObject object.
<a href="#">Release</a> (inherited from <a href="#">TSharedObject</a> )	Decrements the reference count.
<a href="#">SaveToFile</a> (inherited from <a href="#">TBlob</a> )	Saves the contents of the TBlob object to a file.
<a href="#">SaveToStream</a> (inherited from <a href="#">TBlob</a> )	Copies the contents of a TBlob object to a stream.
<a href="#">Size</a> (inherited from <a href="#">TBlob</a> )	Used to learn the size of the TBlob value in bytes.
<a href="#">Truncate</a> (inherited from <a href="#">TBlob</a> )	Sets new TBlob size and discards all data over it.
<a href="#">Write</a> (inherited from <a href="#">TBlob</a> )	Stores a raw sequence of bytes into a TBlob object.

### See Also

- [TCompressedBlob Class](#)
- [TCompressedBlob Class Members](#)

## 17.25.1.3.2.1 Compressed Property

Used to indicate if the Blob is compressed.

**Class**

[TCompressedBlob](#)

**Syntax**

```
property Compressed: boolean;
```

**Remarks**

Indicates whether the Blob is compressed. Set this property to True or False to compress or decompress the Blob.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.3.2.2 CompressedSize Property

Used to indicate compressed size of the Blob data.

**Class**

[TCompressedBlob](#)

**Syntax**

```
property CompressedSize: Cardinal;
```

**Remarks**

Indicates compressed size of the Blob data.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.4 TDBObject Class

A base class for classes that work with user-defined data types that have attributes. For a list of all members of this type, see [TDBObject](#) members.

**Unit**

[MemData](#)

**Syntax**

```
TDBObject = class (TSharedObject);
```

**Remarks**

TDBObject is a base class for classes that work with user-defined data types that have attributes.

**Inheritance Hierarchy**

[TSharedObject](#)

**TDBObject**

---

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.4.1 Members

[TDBObject](#) class overview.

### Properties

Name	Description
<a href="#">RefCount</a> (inherited from <a href="#">TSharedObject</a> )	Used to return the count of reference to a TSharedObject object.

### Methods

Name	Description
<a href="#">AddRef</a> (inherited from <a href="#">TSharedObject</a> )	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">Release</a> (inherited from <a href="#">TSharedObject</a> )	Decrements the reference count.

© 1997-2013 Devart. All Rights Reserved.

### 17.25.1.5 TObjectType Class

This class is not used.

For a list of all members of this type, see [TObjectType](#) members.

### Unit

[MemData](#)

### Syntax

```
TObjectType = class(TSharedObject);
```

### Inheritance Hierarchy

[TSharedObject](#)  
**TObjectType**

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.5.1 Members

[TObjectType](#) class overview.

### Properties

Name	Description
<a href="#">AttributeCount</a>	Used to indicate the number of attributes of type.
<a href="#">Attributes</a>	Used to access separate attributes.

[DataType](#)

Used to indicate the type of object dtObject, dtArray or dtTable.

[RefCount](#) (inherited from [TSharedObject](#))

Used to return the count of reference to a TSharedObject object.

[Size](#)

Used to learn the size of an object instance.

## Methods

### Name

### Description

[AddRef](#) (inherited from [TSharedObject](#))

Increments the reference count for the number of references dependent on the TSharedObject object.

[AttributeByName](#)

Retrieves attribute information for an attribute when only the attribute's name is known.

[FindAttribute](#)

Indicates whether a specified Attribute component is referenced in the TAttributes object.

[Release](#) (inherited from [TSharedObject](#))

Decrements the reference count.

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.5.2 Properties

Properties of the **TObjectType** class.

For a complete list of the **TObjectType** class members, see the [TObjectType Members](#) topic.

## Public

### Name

### Description

[AddRef](#) (inherited from [TSharedObject](#))

Increments the reference count for the number of references dependent on the TSharedObject object.

[AttributeCount](#)

Used to indicate the number of attributes of type.

[Attributes](#)

Used to access separate attributes.

[DataType](#)

Used to indicate the type of object dtObject, dtArray or dtTable.

[RefCount](#) (inherited from [TSharedObject](#))

Used to return the count of reference to a TSharedObject object.

[Release](#) (inherited from [TSharedObject](#))

Decrements the reference count.

[Size](#)

Used to learn the size of an object instance.

### See Also

- [TObjectType Class](#)
- [TObjectType Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.5.2.1 AttributeCount Property

Used to indicate the number of attributes of type.

### Class

[TObjectType](#)

### Syntax

```
property AttributeCount: Integer;
```

### Remarks

Use the AttributeCount property to determine the number of attributes of type.

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.5.2.2 Attributes Property(Indexer)

Used to access separate attributes.

### Class

[TObjectType](#)

### Syntax

```
property Attributes[Index: integer]: TAttribute;
```

#### Parameters

*Index*

Holds the attribute's ordinal position.

### Remarks

Use the Attributes property to access individual attributes. The value of the Index parameter corresponds to the AttributeNo property of TAttribute.

### See Also

- [TAttribute](#)
- [FindAttribute](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.5.2.3 DataType Property

Used to indicate the type of object dtObject, dtArray or dtTable.

**Class**

[TObjectType](#)

**Syntax**

```
property DataType: Word;
```

**Remarks**

Use the DataType property to determine the type of object dtObject, dtArray or dtTable.

**See Also**

- [MemData](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.5.2.4 Size Property

Used to learn the size of an object instance.

**Class**

[TObjectType](#)

**Syntax**

```
property Size: Integer;
```

**Remarks**

Use the Size property to find out the size of an object instance. Size is a sum of all attribute sizes.

**See Also**

- [TAttribute.Size](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.25.1.5.3 Methods

Methods of the **TObjectType** class.

For a complete list of the **TObjectType** class members, see the [TObjectType Members](#) topic.

**Public**

Name	Description
------	-------------



[AddRef](#) (inherited from [TSharedObject](#))

Increments the reference count for the number of references dependent on the TSharedObject object.

[AttributeByName](#)

Retrieves attribute information for an attribute when only the attribute's name is known.

[FindAttribute](#)

Indicates whether a specified Attribute component is referenced in the TAttributes object.

[RefCount](#) (inherited from [TSharedObject](#))

Used to return the count of reference to a TSharedObject object.

[Release](#) (inherited from [TSharedObject](#))

Decrements the reference count.

### See Also

- [TObjectType Class](#)
- [TObjectType Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.5.3.1 AttributeByName Method

Retrieves attribute information for an attribute when only the attribute's name is known.

### Class

[TObjectType](#)

### Syntax

```
function AttributeByName (Name: string): TAttribute;
```

#### Parameters

*Name*

Holds the name of an existing attribute.

#### Return Value

a TAttribute object for the specified attribute. Otherwise an exception is raised.

### Remarks

Call the AttributeByName method to retrieve attribute information for an attribute when only the attribute's name is known. Name is the name of an existing Attribute. AttributeByName returns a TAttribute object for the specified attribute. If the attribute can not be found, an exception is raised.

### See Also

- [TAttribute](#)
  - [FindAttribute](#)
  - [Attributes](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.5.3.2 FindAttribute Method

Indicates whether a specified Attribute component is referenced in the TAttributes object.

### Class

[TObjectType](#)

### Syntax

```
function FindAttribute (Name: string) : TAttribute;
```

#### Parameters

*Name*

Holds the name of the attribute to search for.

#### Return Value

TAttribute, if an attribute with a matching name was found. Nil Otherwise.

### Remarks

Call FindAttribute to determine if a specified Attribute component is referenced in the TAttributes object. Name is the name of the Attribute for which to search. If FindAttribute finds an Attribute with a matching name, it returns the TAttribute. Otherwise it returns nil.

### See Also

- [TAttribute](#)
  - [AttributeByName](#)
  - [Attributes](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.6 TSharedObject Class

A base class that allows to simplify memory management for object referenced by several other objects.

For a list of all members of this type, see [TSharedObject](#) members.

### Unit

[MemData](#)

### Syntax

```
TSharedObject = class (System.TObject) ;
```

### Remarks

TSharedObject allows to simplify memory management for object referenced by several other objects. TSharedObject holds a count of references to itself. When any object (referer object) is going to use TSharedObject, it calls the TSharedObject.AddRef method. Referer object has to call the TSharedObject.Release method after using TSharedObject.

## See Also

- [TBlob](#)
- [TObjectType](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.25.1.6.1 Members

[TSharedObject](#) class overview.

## Properties

Name	Description
<a href="#">RefCount</a>	Used to return the count of reference to a TSharedObject object.

## Methods

Name	Description
<a href="#">AddRef</a>	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">Release</a>	Decrements the reference count.

© 1997-2013 Devart. All Rights Reserved.

### 17.25.1.6.2 Properties

Properties of the **TSharedObject** class.  
For a complete list of the **TSharedObject** class members, see the [TSharedObject Members](#) topic.

## Public

Name	Description
<a href="#">RefCount</a>	Used to return the count of reference to a TSharedObject object.

## See Also

- [TSharedObject Class](#)
- [TSharedObject Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.6.2.1 RefCount Property

Used to return the count of reference to a TSharedObject object.

#### Class

[TSharedObject](#)

#### Syntax

```
property RefCount: Integer;
```

#### Remarks

Returns the count of reference to a TSharedObject object.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.6.3 Methods

Methods of the **TSharedObject** class.

For a complete list of the **TSharedObject** class members, see the [TSharedObject Members](#) topic.

#### Public

Name	Description
<a href="#">AddRef</a>	Increments the reference count for the number of references dependent on the TSharedObject object.
<a href="#">Release</a>	Decrements the reference count.

#### See Also

- [TSharedObject Class](#)
  - [TSharedObject Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.1.6.3.1 AddRef Method

Increments the reference count for the number of references dependent on the TSharedObject object.

#### Class

[TSharedObject](#)

#### Syntax

```
procedure AddRef;
```

#### Remarks

Increments the reference count for the number of references dependent on the

TSharedObject object.

## See Also

- [Release](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.25.1.6.3.2 Release Method

Decrements the reference count.

## Class

[TSharedObject](#)

## Syntax

```
procedure Release;
```

## Remarks

Call the Release method to decrement the reference count. When RefCount is 1, TSharedObject is deleted from memory.

## See Also

- [AddRef](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.25.2 Types

Types in the **MemData** unit.

## Types

Name	Description
<a href="#">TLocateExOptions</a>	Represents the set of <a href="#">TLocateExOption</a> .
<a href="#">TUpdateReckinds</a>	Represents the set of TUpdateReckind.

© 1997-2013 Devart. All Rights Reserved.

### 17.25.2.1 TLocateExOptions Set

Represents the set of [TLocateExOption](#).

## Unit

[MemData](#)

## Syntax

---

```
TLocateExOptions = set of TLocateExOption;
```

---

© 1997-2013 Devart. All Rights Reserved.

### 17.25.2 TUpdateRecKinds Set

Represents the set of TUpdateRecKind.

#### Unit

[MemData](#)

#### Syntax

```
TUpdateRecKinds = set of TUpdateRecKind;
```

---

© 1997-2013 Devart. All Rights Reserved.

### 17.25.3 Enumerations

Enumerations in the **MemData** unit.

#### Enumerations

Name	Description
<a href="#">TCompressBlobMode</a>	Specifies when the values should be compressed and the way they should be stored.
<a href="#">TConnLostCause</a>	Specifies the cause of the connection loss.
<a href="#">TDANumericType</a>	Specifies the format of storing and representing of the NUMERIC (DECIMAL) fields.
<a href="#">TLocateExOption</a>	Allows to set additional search parameters which will be used by the LocateEx method.
<a href="#">TSortType</a>	Specifies a sort type for string fields.
<a href="#">TUpdateRecKind</a>	Indicates records for which the ApplyUpdates method will be performed.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.3.1 TCompressBlobMode Enumeration

Specifies when the values should be compressed and the way they should be stored.

#### Unit

[MemData](#)

**Syntax**

```
TCompressBlobMode = (cbNone, cbClient, cbServer, cbClientServer);
```

**Values**

Value	Meaning
<b>cbClient</b>	Values are compressed and stored as compressed data at the client side. Before posting data to the server decompression is performed and data at the server side stored in the original form. Allows to reduce used client memory due to increase access time to field values. The time spent on the opening DataSet and executing Post increases.
<b>cbClientServer</b>	Values are compressed and stored in compressed form. Allows to decrease the volume of used memory at client and server sides. Access time to the field values increases as for cbClient. The time spent on opening DataSet and executing Post decreases. <b>Note:</b> On using cbServer or cbClientServer data on the server is stored as compressed. Other applications can add records in uncompressed format but can't read and write already compressed data. If compressed BLOB is partially changed by another application (if signature was not changed), DAC will consider its value as NULL. Blob compression is not applied to Memo fields because of possible cutting.
<b>cbNone</b>	Values not compressed. The default value.
<b>cbServer</b>	Values are compressed before passing to the server and store at the server in compressed form. Allows to decrease database size on the server. Access time to the field values does not change. The time spent on opening DataSet and executing Post usually decreases.

© 1997-2013 Devart. All Rights Reserved.

**17.25.3.2 TConnLostCause Enumeration**

Specifies the cause of the connection loss.

**Unit**

[MemData](#)

**Syntax**

```
TConnLostCause = (clUnknown, clExecute, clOpen, clRefresh,
  clApply, clServiceQuery, clTransStart, clConnectionApply,
  clConnect);
```

**Values**

Value	Meaning
<b>clApply</b>	Connection loss detected during DataSet.ApplyUpdates (Reconnect/Reexecute possible).

<b>clConnect</b>	Connection loss detected during connection establishing (Reconnect possible).
<b>clConnectionApply</b>	Connection loss detected during Connection.ApplyUpdates (Reconnect/Reexecute possible).
<b>clExecute</b>	Connection loss detected during SQL execution (Reconnect with exception is possible).
<b>clOpen</b>	Connection loss detected during execution of a SELECT statement (Reconnect with exception possible).
<b>clRefresh</b>	Connection loss detected during query opening (Reconnect/Reexecute possible).
<b>clServiceQuery</b>	Connection loss detected during service information request (Reconnect/Reexecute possible).
<b>clTransStart</b>	Connection loss detected during transaction start (Reconnect/Reexecute possible). clTransStart has less priority then clConnectionApply.
<b>clUnknown</b>	The connection loss reason is unknown.

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.3.3 TDANumericType Enumeration

Specifies the format of storing and representing of the NUMERIC (DECIMAL) fields.

##### Unit

[MemData](#)

##### Syntax

```
TDANumericType = (ntFloat, ntBCD, ntFmtBCD);
```

##### Values

Value	Meaning
<b>ntBCD</b>	Data is stored on the client side as currency and represented as TBCDField. This format allows storing data with precision up to 0,0001.
<b>ntFloat</b>	Data stored on the client side is in double format and represented as TFloatField. The default value.
<b>ntFmtBCD</b>	Data is represented as TFMTBCDField. TFMTBCDField gives greater precision and accuracy than TBCDField, but it is slower. Not supported for Delphi 5 and C++Builder 5.

© 1997-2013 Devart. All Rights Reserved.

#### 17.25.3.4 TLocateExOption Enumeration

Allows to set additional search parameters which will be used by the LocateEx method.

##### Unit

[MemData](#)



## Syntax

```
TLocateExOption = (lxCaseInsensitive, lxPartialKey, lxNearest,  
    lxNext, lxUp, lxPartialCompare);
```

## Values

Value	Meaning
<b>lxCaseInsensitive</b>	Similar to loCaseInsensitive. Key fields and key values are matched without regard to the case.
<b>lxNearest</b>	LocateEx moves the cursor to a specific record in a dataset or to the first record in the dataset that is greater than the values specified in the KeyValues parameter. For this option to work correctly dataset should be sorted by the fields the search is performed in. If dataset is not sorted, the function may return a line that is not connected with the search condition.
<b>lxNext</b>	LocateEx searches from the current record.
<b>lxPartialCompare</b>	Similar to lxPartialKey, but the difference is that it can process value entries in any position. For example, 'HAM' would match both 'HAMM', 'HAMMER.', and also 'MR HAMMER'.
<b>lxPartialKey</b>	Similar to loPartialKey. Key values can include only a part of the matching key field value. For example, 'HAM' would match both 'HAMM' and 'HAMMER.', but not 'MR HAMMER'.
<b>lxUp</b>	LocateEx searches from the current record to the first record.

© 1997-2013 Devart. All Rights Reserved.

### 17.25.3.5 TSortType Enumeration

Specifies a sort type for string fields.

## Unit

[MemData](#)

## Syntax

```
TSortType = (stCaseSensitive, stCaseInsensitive, stBinary);
```

## Values

Value	Meaning
<b>stBinary</b>	Sorting by character ordinal values (this comparison is also case sensitive).
<b>stCaseInsensitive</b>	Sorting without case sensitivity.
<b>stCaseSensitive</b>	Sorting with case sensitivity.

© 1997-2013 Devart. All Rights Reserved.

### 17.25.3.6 TUpdateRecKind Enumeration

Indicates records for which the ApplyUpdates method will be performed.

#### Unit

[MemData](#)

#### Syntax

```
TUpdateRecKind = (ukUpdate, ukInsert, ukDelete);
```

#### Values

Value	Meaning
<b>ukDelete</b>	<a href="#">ApplyUpdates</a> will be performed for deleted records.
<b>ukInsert</b>	<a href="#">ApplyUpdates</a> will be performed for inserted records.
<b>ukUpdate</b>	<a href="#">ApplyUpdates</a> will be performed for updated records.

© 1997-2013 Devart. All Rights Reserved.

## 17.26 MemDS

This unit contains implementation of the TMemDataSet class.

#### Classes

Name	Description
<a href="#">TMemDataSet</a>	A base class for working with data and manipulating data in memory.

#### Variables

Name	Description
<a href="#">DoNotRaiseExcetionOnUaFail</a>	An exception will be raised if the value of the UpdateAction parameter is uaFail.
<a href="#">SendDataSetChangeEventAfterOpen</a>	The DataSetChange event is sent after a dataset gets open. It was necessary to fix a problem with disappeared vertical scrollbar in some types of DB-aware grids.

© 1997-2013 Devart. All Rights Reserved.

### 17.26.1 Classes

Classes in the **MemDS** unit.

#### Classes

Name	Description
------	-------------

[TMemDataSet](#)

A base class for working with data and manipulating data in memory.

© 1997-2013 Devart. All Rights Reserved.

**17.26.1.1 TMemDataSet Class**

A base class for working with data and manipulating data in memory.  
For a list of all members of this type, see [TMemDataSet](#) members.

**Unit**

[MemDS](#)

**Syntax**

```
TMemDataSet = class (TDataSet);
```

**Remarks**

TMemDataSet derives from the TDataSet database-engine independent set of properties, events, and methods for working with data and introduces additional techniques to store and manipulate data in memory.

© 1997-2013 Devart. All Rights Reserved.

**17.26.1.1.1 Members**

[TMemDataSet](#) class overview.

**Properties**

Name	Description
<a href="#">CachedUpdates</a>	Used to enable or disable the use of cached updates for a dataset.
<a href="#">IndexFieldNames</a>	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">LocalConstraints</a>	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a>	Used to prevent implicit update of rows on database server.
<a href="#">Prepared</a>	Determines whether a query is prepared for execution or not.
<a href="#">UpdateRecordTypes</a>	Used to indicate the update status for the current record when cached updates are enabled.

[UpdatesPending](#)

Used to check the status of the cached updates buffer.

## Methods

Name	Description
<a href="#">ApplyUpdates</a>	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">CancelUpdates</a>	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a>	Clears the cached updates buffer.
<a href="#">DeferredPost</a>	Makes permanent changes to the database server.
<a href="#">GetBlob</a>	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">Locate</a>	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a>	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">Prepare</a>	Allocates resources and creates field components for a dataset.
<a href="#">RestoreUpdates</a>	Marks all records in the cache of updates as unapplied.
<a href="#">RevertRecord</a>	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveToXML</a>	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">UnPrepare</a>	Frees the resources allocated for a previously prepared query on the server and client sides.

[UpdateResult](#)

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled. Indicates the current update status for the dataset when cached updates are enabled.

[UpdateStatus](#)

## Events

Name	Description
<a href="#">OnUpdateError</a>	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a>	Occurs when a single update component can not handle the updates.

© 1997-2013 Devart. All Rights Reserved.

### 17.26.1.1.2 Properties

Properties of the **TMemDataSet** class.

For a complete list of the **TMemDataSet** class members, see the [TMemDataSet Members](#) topic.

## Public

Name	Description
<a href="#">CachedUpdates</a>	Used to enable or disable the use of cached updates for a dataset.
<a href="#">IndexFieldNames</a>	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">LocalConstraints</a>	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a>	Used to prevent implicit update of rows on database server.
<a href="#">Prepared</a>	Determines whether a query is prepared for execution or not.
<a href="#">UpdateRecordTypes</a>	Used to indicate the update status for the current record when cached updates are enabled.

[UpdatesPending](#)

Used to check the status of the cached updates buffer.

**See Also**

- [TMemDataSet Class](#)
  - [TMemDataSet Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.2.1 CachedUpdates Property

Used to enable or disable the use of cached updates for a dataset.

**Class**

[TMemDataSet](#)

**Syntax**

```
property CachedUpdates: boolean default False;
```

**Remarks**

Use the CachedUpdates property to enable or disable the use of cached updates for a dataset. Setting CachedUpdates to True enables updates to a dataset (such as posting changes, inserting new records, or deleting records) to be stored in an internal cache on the client side instead of being written directly to the dataset's underlying database tables. When changes are completed, an application writes all cached changes to the database in the context of a single transaction.

Cached updates are especially useful for client applications working with remote database servers. Enabling cached updates brings up the following benefits:

- Fewer transactions and shorter transaction times.
- Minimized network traffic.

The potential drawbacks of enabling cached updates are:

- Other applications can access and change the actual data on the server while users are editing local copies of data, resulting in an update conflict when cached updates are applied to the database.
- Other applications cannot access data changes made by an application until its cached updates are applied to the database.

The default value is False.

**Note:** When establishing master/detail relationship the CachedUpdates property of detail dataset works properly only when [TCustomDADataset.Options](#) is set to True.

**See Also**

- [UpdatesPending](#)
  - [TMemDataSet.ApplyUpdates](#)
  - [RestoreUpdates](#)
  - [CommitUpdates](#)
  - [CancelUpdates](#)
  - [UpdateStatus](#)
  - [TCustomDADataset.Options](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.2.2 IndexFieldNames Property

Used to get or set the list of fields on which the recordset is sorted.

**Class**

[TMemDataSet](#)

**Syntax**

```
property IndexFieldNames: string;
```

**Remarks**

Use the IndexFieldNames property to get or set the list of fields on which the recordset is sorted. Specify the name of each column in IndexFieldNames to use as an index for a table. Ordering of column names is significant. Separate names with semicolon. The specified columns don't need to be indexed. Set IndexFieldNames to an empty string to reset the recordset to the sort order originally used when the recordset's data was first retrieved.

Each field may optionally be followed by the keyword ASC / DESC or CIS / CS / BIN. Use ASC, DESC keywords to specify a sort direction for the field. If one of these keywords is not used, the default sort direction for the field is ascending.

Use CIS, CS or BIN keywords to specify a sort type for string fields:

CIS - compare without case sensitivity;

CS - compare with case sensitivity;

BIN - compare by character ordinal values (this comparison is also case sensitive).

If a dataset uses a [TCustomDAConnection](#) component, the default value of sort type depends on the [TCustomDAConnection.Options](#) option of the connection. If a dataset does not use a connection ([TVirtualTable](#) dataset), the default is CS.

Read IndexFieldNames to determine the field (or fields) on which the recordset is sorted.

Ordering is processed locally.

**Note:** You cannot process ordering by BLOB fields.

**Example**

The following procedure illustrates how to set IndexFieldNames in response to a button click:

```
DataSet1.IndexFieldNames := 'LastName ASC CIS; DateDue DESC';
```

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.2.3 LocalConstraints Property

Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

**Class**

[TMemDataSet](#)

**Syntax**

```
property LocalConstraints: boolean default True;
```

**Remarks**

Use the LocalConstraints property to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet. When LocalConstraints is True, TMemDataSet ignores NOT NULL server constraints. It is useful for tables that have fields updated by triggers. LocalConstraints is obsolete, and is only included for backward compatibility. The default value is True.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.2.4 LocalUpdate Property

Used to prevent implicit update of rows on database server.

**Class**

[TMemDataSet](#)

**Syntax**

```
property LocalUpdate: boolean default False;
```

**Remarks**

Set the LocalUpdate property to True to prevent implicit update of rows on database server. Data changes are cached locally in client memory.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.2.5 Prepared Property

Determines whether a query is prepared for execution or not.

**Class**

[TMemDataSet](#)

**Syntax**

```
property Prepared: boolean;
```

**Remarks**

Determines whether a query is prepared for execution or not.

**See Also**

- [Prepare](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.2.6 UpdateRecordTypes Property

Used to indicate the update status for the current record when cached updates are enabled.



## Class

[TMemDataSet](#)

## Syntax

```
property UpdateRecordTypes: TUpdateRecordTypes default  
[rtModified, rtInserted, rtUnmodified];
```

## Remarks

Use the UpdateRecordTypes property to determine the update status for the current record when cached updates are enabled. Update status can change frequently as records are edited, inserted, or deleted. UpdateRecordTypes offers a convenient method for applications to assess the current status before undertaking or completing operations that depend on the update status of records.

## See Also

- [CachedUpdates](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.26.1.1.2.7 UpdatesPending Property

Used to check the status of the cached updates buffer.

## Class

[TMemDataSet](#)

## Syntax

```
property UpdatesPending: boolean;
```

## Remarks

Use the UpdatesPending property to check the status of the cached updates buffer. If UpdatesPending is True, then there are edited, deleted, or inserted records remaining in local cache and not yet applied to the database. If UpdatesPending is False, there are no such records in the cache.

## See Also

- [CachedUpdates](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.26.1.1.3 Methods

Methods of the **TMemDataSet** class.  
For a complete list of the **TMemDataSet** class members, see the [TMemDataSet Members](#) topic.

**Public**

<b>Name</b>	<b>Description</b>
<a href="#">ApplyUpdates</a>	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">CancelUpdates</a>	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a>	Clears the cached updates buffer.
<a href="#">DeferredPost</a>	Makes permanent changes to the database server.
<a href="#">GetBlob</a>	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">Locate</a>	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a>	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">Prepare</a>	Allocates resources and creates field components for a dataset.
<a href="#">RestoreUpdates</a>	Marks all records in the cache of updates as unapplied.
<a href="#">RevertRecord</a>	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveToXML</a>	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">UnPrepare</a>	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateResult</a>	Reads the status of the latest call to the <a href="#">ApplyUpdates</a> method while cached updates are enabled.

[UpdateStatus](#)

Indicates the current update status for the dataset when cached updates are enabled.

**See Also**

- [TMemDataSet Class](#)
- [TMemDataSet Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.3.1 ApplyUpdates Method

Writes dataset's pending cached updates to a database.

**Class**

[TMemDataSet](#)

**Overload List**

Name	Description
<a href="#">ApplyUpdates</a>	Writes dataset's pending cached updates to a database.
<a href="#">ApplyUpdates(const UpdateReckinds: TUpdateReckinds)</a>	Writes dataset's pending cached updates of specified records to a database.

© 1997-2013 Devart. All Rights Reserved.

Writes dataset's pending cached updates to a database.

**Class**

[TMemDataSet](#)

**Syntax**

```
procedure ApplyUpdates; overload; virtual
```

**Remarks**

Call the ApplyUpdates method to write a dataset's pending cached updates to a database. This method passes cached data to the database, but the changes are not committed to the database if there is an active transaction. An application must explicitly call the database component's Commit method to commit the changes to the database if the write is successful, or call the database's Rollback method to undo the changes if there is an error.

Following a successful write to the database, and following a successful call to a connection's Commit method, an application should call the CommitUpdates method to clear the cached update buffer.

**Note:** The preferred method for updating datasets is to call a connection component's ApplyUpdates method rather than to call each individual dataset's ApplyUpdates method. The connection component's ApplyUpdates method takes care of committing and rolling back transactions and clearing the cache when the operation is successful.

## See Also

- [TMemDataSet.CachedUpdates](#)
  - [TMemDataSet.CancelUpdates](#)
  - [TMemDataSet.CommitUpdates](#)
  - [TMemDataSet.UpdateStatus](#)
- 

© 1997-2013 Devart. All Rights Reserved.

Writes dataset's pending cached updates of specified records to a database.

## Class

[TMemDataSet](#)

## Syntax

```
procedure ApplyUpdates(const UpdateRecKinds: TUpdateRecKinds);  
overload; virtual
```

### Parameters

*UpdateRecKinds*

Indicates records for which the ApplyUpdates method will be performed.

## Remarks

Call the ApplyUpdates method to write a dataset's pending cached updates of specified records to a database. This method passes cached data to the database, but the changes are not committed to the database if there is an active transaction. An application must explicitly call the database component's Commit method to commit the changes to the database if the write is successful, or call the database's Rollback method to undo the changes if there is an error. Following a successful write to the database, and following a successful call to a connection's Commit method, an application should call the CommitUpdates method to clear the cached update buffer.

**Note:** The preferred method for updating datasets is to call a connection component's ApplyUpdates method rather than to call each individual dataset's ApplyUpdates method. The connection component's ApplyUpdates method takes care of committing and rolling back transactions and clearing the cache when the operation is successful.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.26.1.1.3.2 CancelUpdates Method

Clears all pending cached updates from cache and restores dataset in its prior state.

## Class

[TMemDataSet](#)

## Syntax

```
procedure CancelUpdates;
```

## Remarks

Call the CancelUpdates method to clear all pending cached updates from cache and restore dataset in its prior state.

It restores the dataset to the state it was in when the table was opened, cached updates were last enabled, or updates were last successfully applied to the database.

When a dataset is closed, or the CachedUpdates property is set to False, CancelUpdates is called automatically.

## See Also

- [CachedUpdates](#)
- [TMemDataSet.ApplyUpdates](#)
- [UpdateStatus](#)

---

© 1997-2013 Devart. All Rights Reserved.

### 17.26.1.1.3.3 CommitUpdates Method

Clears the cached updates buffer.

## Class

[TMemDataSet](#)

## Syntax

```
procedure CommitUpdates;
```

## Remarks

Call the CommitUpdates method to clear the cached updates buffer after both a successful call to ApplyUpdates and a database component's Commit method.

Clearing the cache after applying updates ensures that the cache is empty except for records that could not be processed and were skipped by the OnUpdateRecord or OnUpdateError event handlers. An application can attempt to modify the records still in cache.

CommitUpdates also checks whether there are pending updates in dataset. And if there are, it calls ApplyUpdates.

Record modifications made after a call to CommitUpdates repopulate the cached update buffer and require a subsequent call to ApplyUpdates to move them to the database.

## See Also

- [CachedUpdates](#)
- [TMemDataSet.ApplyUpdates](#)
- [UpdateStatus](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.3.4 DeferredPost Method

Makes permanent changes to the database server.

**Class**

[TMemDataSet](#)

**Syntax**

```
procedure DeferredPost;
```

**Remarks**

Call DeferredPost to make permanent changes to the database server while retaining dataset in its state whether it is dsEdit or dsInsert. Explicit call to the Cancel method after DeferredPost has been applied does not abandon modifications to a dataset already fixed in database.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.3.5 GetBlob Method

Retrieves TBlob object for a field or current record when only its name or the field itself is known.

**Class**

[TMemDataSet](#)

**Overload List**

Name	Description
<a href="#">GetBlob(Field: TField)</a>	Retrieves TBlob object for a field or current record when the field itself is known.
<a href="#">GetBlob(const FieldName: string)</a>	Retrieves TBlob object for a field or current record when its name is known.

---

© 1997-2013 Devart. All Rights Reserved.

Retrieves TBlob object for a field or current record when the field itself is known.

**Class**

[TMemDataSet](#)

**Syntax**

```
function GetBlob(Field: TField): TBlob; overload
```

**Parameters***Field*

Holds an existing TField object.

**Return Value**

TBlob object that was retrieved.

## Remarks

Call the GetBlob method to retrieve TBlob object for a field or current record when only its name or the field itself is known. FieldName is the name of an existing field. The field should have MEMO or BLOB type.

© 1997-2013 Devart. All Rights Reserved.

Retrieves TBlob object for a field or current record when its name is known.

## Class

[TMemDataSet](#)

## Syntax

```
function GetBlob(const FieldName: string): TBlob; overload
```

### Parameters

*FieldName*

Holds the name of an existing field.

### Return Value

TBlob object that was retrieved.

## Example

```
IBCQuery1.GetBlob('Comment').SaveToFile('Comment.txt');
```

## See Also

- 

[TBlob](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.26.1.1.3.6 Locate Method

Searches a dataset for a specific record and positions the cursor on it.

## Class

[TMemDataSet](#)

## Overload List

Name	Description
<a href="#">Locate(const KeyFields: array of TField; const KeyValues: variant; Options: TLocateOptions)</a>	Searches a dataset by the specified fields for a specific record and positions cursor on it.
<a href="#">Locate(const KeyFields: string; const KeyValues: variant; Options: TLocateOptions)</a>	Searches a dataset by the fields specified by name for a specific record and positions the cursor on it.

© 1997-2013 Devart. All Rights Reserved.

Searches a dataset by the specified fields for a specific record and positions cursor on it.

## Class

[TMemDataSet](#)

## Syntax

```
function Locate(const KeyFields: array of TField; const KeyValues:  
variant; Options: TLocateOptions): boolean; reintroduce;  
overload
```

### Parameters

#### *KeyFields*

Holds TField objects in which to search.

#### *KeyValues*

Holds the variant that specifies the values to match in the key fields.

#### *Options*

Holds additional search latitude when searching in string fields.

### Return Value

True if it finds a matching record, and makes this record the current one.  
Otherwise it returns False.

---

© 1997-2013 Devart. All Rights Reserved.

Searches a dataset by the fields specified by name for a specific record and positions the cursor on it.

## Class

[TMemDataSet](#)

## Syntax

```
function Locate(const KeyFields: string; const KeyValues: variant;  
Options: TLocateOptions): boolean; overload; override
```

### Parameters

#### *KeyFields*

Holds a semicolon-delimited list of field names in which to search.

#### *KeyValues*

Holds the variant that specifies the values to match in the key fields.

#### *Options*

Holds additional search latitude when searching in string fields.

### Return Value

True if it finds a matching record, and makes this record the current one.  
Otherwise it returns False.

## Remarks



Call the Locate method to search a dataset for a specific record and position cursor on it.

KeyFields is a string containing a semicolon-delimited list of field names on which to search.

KeyValues is a variant that specifies the values to match in the key fields. If KeyFields lists a single field, KeyValues specifies the value for that field on the desired record. To specify multiple search values, pass a variant array as KeyValues, or construct a variant array on the fly using the VarArrayOf routine. An example is provided below.

Options is a set that optionally specifies additional search latitude when searching in string fields. If Options contains the loCaseInsensitive setting, then Locate ignores case when matching fields. If Options contains the loPartialKey setting, then Locate allows partial-string matching on strings in KeyValues. If Options is an empty set, or if KeyFields does not include any string fields, Options is ignored.

Locate returns True if it finds a matching record, and makes this record the current one. Otherwise it returns False.

The Locate function works faster when dataset is locally sorted on the KeyFields fields. Local dataset sorting can be set with the [TMemDataSet.IndexFieldNames](#) property.

## Example

An example of specifying multiple search values:

```
with CustTable do
    Locate('Company;Contact;Phone', VarArrayOf(['Sight Diver', 'P',
        '408-431-1000']), [loPartialKey]);
```

## See Also

- [TMemDataSet.IndexFieldNames](#)
- [TMemDataSet.LocateEx](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.26.1.1.3.7 LocateEx Method

Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet.

## Class

[TMemDataSet](#)

## Overload List

Name	Description
<a href="#">LocateEx(const KeyFields: array of TField; const KeyValues: variant; Options: TLocateExOptions)</a>	Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet by the specified fields.

[LocateEx\(const KeyFields: string; const KeyValues: variant; Options: TLocateExOptions\)](#)

Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet by the specified field names.

© 1997-2013 Devart. All Rights Reserved.

Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet by the specified fields.

## Class

[TMemDataSet](#)

## Syntax

```
function LocateEx(const KeyFields: array of TField; const
  KeyValues: variant; Options: TLocateExOptions): boolean;
overload
```

### Parameters

*KeyFields*

Holds TField objects to search in.

*KeyValues*

Holds the values of the fields to search for.

*Options*

Holds additional search parameters which will be used by the LocateEx method.

### Return Value

True, if a matching record was found. Otherwise returns False.

© 1997-2013 Devart. All Rights Reserved.

Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet by the specified field names.

## Class

[TMemDataSet](#)

## Syntax

```
function LocateEx(const KeyFields: string; const KeyValues:
  variant; Options: TLocateExOptions): boolean; overload
```

### Parameters

*KeyFields*

Holds the fields to search in.

*KeyValues*

Holds the values of the fields to search for.

*Options*

Holds additional search parameters which will be used by the LocateEx method.

### Return Value

True, if a matching record was found. Otherwise returns False.

### Remarks

Call the LocateEx method when you need some features not to be included to the [TMemDataSet.Locate](#) method of TDataSet.

LocateEx returns True if it finds a matching record, and makes that record the current one. Otherwise LocateEx returns False.

The LocateEx function works faster when dataset is locally sorted on the KeyFields fields. Local dataset sorting can be set with the [TMemDataSet.IndexFieldNames](#) property.

**Note:** Please add the MemData unit to the "uses" list to use the TLocalExOption enumeration.

### See Also

- [TMemDataSet.IndexFieldNames](#)
- [TMemDataSet.Locate](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.26.1.1.3.8 Prepare Method

Allocates resources and creates field components for a dataset.

### Class

[TMemDataSet](#)

### Syntax

```
procedure Prepare; virtual;
```

### Remarks

Call the Prepare method to allocate resources and create field components for a dataset. To learn whether dataset is prepared or not use the Prepared property.

The UnPrepare method unprepares a query.

**Note:** When you change the text of a query at runtime, the query is automatically closed and unprepared.

The Prepare method is called automatically by the Open method if the dataset is not prepared.

### See Also

- [Prepared](#)
- [UnPrepare](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.3.9 RestoreUpdates Method

Marks all records in the cache of updates as unapplied.

**Class**

[TMemDataSet](#)

**Syntax**

```
procedure RestoreUpdates;
```

**Remarks**

Call the RestoreUpdates method to return the cache of updates to its state before calling ApplyUpdates. RestoreUpdates marks all records in the cache of updates as unapplied. It is useful when ApplyUpdates fails.

**See Also**

- [CachedUpdates](#)
  - [TMemDataSet.ApplyUpdates](#)
  - [CancelUpdates](#)
  - [UpdateStatus](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.3.10 RevertRecord Method

Cancels changes made to the current record when cached updates are enabled.

**Class**

[TMemDataSet](#)

**Syntax**

```
procedure RevertRecord;
```

**Remarks**

Call the RevertRecord method to undo changes made to the current record when cached updates are enabled.

**See Also**

- [CachedUpdates](#)
  - [CancelUpdates](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.3.11 SaveToXML Method

Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

## Class

[TMemDataSet](#)

## Overload List

Name	Description
<a href="#">SaveToXML(Destination: TStream)</a>	Saves the current dataset data to a stream in the XML format compatible with ADO format.
<a href="#">SaveToXML(const FileName: string)</a>	Saves the current dataset data to a file in the XML format compatible with ADO format.

© 1997-2013 Devart. All Rights Reserved.

Saves the current dataset data to a stream in the XML format compatible with ADO format.

## Class

[TMemDataSet](#)

## Syntax

```
procedure SaveToXML(Destination: TStream); overload
```

### Parameters

*Destination*

Holds a TStream object.

## Remarks

Call the SaveToXML method to save the current dataset data to a file or a stream in the XML format compatible with ADO format.

If the destination file already exists, it is overwritten. It remains open from the first call to SaveToXML until the dataset is closed. This file can be read by other applications while it is opened, but they cannot write to the file.

When saving data to a stream, a TStream object must be created and its position must be set in a preferable value.

## See Also

- [TVirtualTable.LoadFromFile](#)
- [TVirtualTable.LoadFromStream](#)

© 1997-2013 Devart. All Rights Reserved.

Saves the current dataset data to a file in the XML format compatible with ADO format.

## Class

[TMemDataSet](#)

## Syntax

```
procedure SaveToXML(const FileName: string); overload
```

### Parameters

*FileName*

Holds the name of a destination file.

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.26.1.1.3.12 UnPrepare Method

Frees the resources allocated for a previously prepared query on the server and client sides.

## Class

[TMemDataSet](#)

## Syntax

```
procedure UnPrepare; virtual;
```

## Remarks

Call the UnPrepare method to free the resources allocated for a previously prepared query on the server and client sides.

**Note:** When you change the text of a query at runtime, the query is automatically closed and unprepared.

## See Also

- [Prepare](#)
- 

© 1997-2013 Devart. All Rights Reserved.

#### 17.26.1.1.3.13 UpdateResult Method

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

## Class

[TMemDataSet](#)

## Syntax

```
function UpdateResult: TUpdateAction;
```

### Return Value

a value of the TUpdateAction enumeration.

## Remarks

Call the UpdateResult method to read the status of the latest call to the ApplyUpdates method while cached updates are enabled. UpdateResult reflects updates made on the records that have been edited, inserted, or deleted.

UpdateResult works on the record by record basis and is applicable to the current record only.

## See Also

- [CachedUpdates](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.26.1.1.3.14 UpdateStatus Method

Indicates the current update status for the dataset when cached updates are enabled.

## Class

[TMemDataSet](#)

## Syntax

```
function UpdateStatus: TUpdateStatus; override;  
Return Value
```

a value of the TUpdateStatus enumeration.

## Remarks

Call the UpdateStatus method to determine the current update status for the dataset when cached updates are enabled. Update status can change frequently as records are edited, inserted, or deleted. UpdateStatus offers a convenient method for applications to assess the current status before undertaking or completing operations that depend on the update status of the dataset.

## See Also

- [CachedUpdates](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.26.1.1.4 Events

Events of the **TMemDataSet** class.

For a complete list of the **TMemDataSet** class members, see the [TMemDataSet Members](#) topic.

## Public

Name	Description
<a href="#">OnUpdateError</a>	Occurs when an exception is generated while cached updates are applied to a database.

[OnUpdateRecord](#)

Occurs when a single update component can not handle the updates.

**See Also**

- [TMemDataSet Class](#)
  - [TMemDataSet Class Members](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.26.1.1.4.1 OnUpdateError Event

Occurs when an exception is generated while cached updates are applied to a database.

**Class**

[TMemDataSet](#)

**Syntax**

```
property OnUpdateError: TUpdateErrorEvent;
```

**Remarks**

Write the OnUpdateError event handler to respond to exceptions generated when cached updates are applied to a database.

E is a pointer to an EDatabaseError object from which application can extract an error message and the actual cause of the error condition. The OnUpdateError handler can use this information to determine how to respond to the error condition. UpdateKind describes the type of update that generated the error.

UpdateAction indicates the action to take when the OnUpdateError handler exits. On entry into the handler, UpdateAction is always set to uaFail. If OnUpdateError can handle or correct the error, set UpdateAction to uaRetry before exiting the error handler.

The error handler can use the TField.OldValue and TField.NewValue properties to evaluate error conditions and set TField.NewValue to a new value to reapply. In this case, set UpdateAction to uaRetry before exiting.

**Note:** If a call to ApplyUpdates raises an exception and ApplyUpdates is not called within the context of a try...except block, an error message is displayed. If the OnUpdateError handler cannot correct the error condition and leaves UpdateAction set to uaFail, the error message is displayed twice. To prevent redisplay, set UpdateAction to uaAbort in the error handler.

**See Also**

- [CachedUpdates](#)
- 

© 1997-2013 Devart. All Rights Reserved.



## 17.26.1.1.4.2 OnUpdateRecord Event

Occurs when a single update component can not handle the updates.

**Class**

[TMemDataSet](#)

**Syntax**

```
property OnUpdateRecord: TUpdateRecordEvent;
```

**Remarks**

Write the OnUpdateRecord event handler to process updates that cannot be handled by a single update component, such as implementation of cascading updates, insertions, or deletions. This handler is also useful for applications that require additional control over parameter substitution in update components.

UpdateKind describes the type of update to perform.

UpdateAction indicates the action taken by the OnUpdateRecord handler before it exits. On entry into the handler, UpdateAction is always set to uaFail. If

OnUpdateRecord is successful, it should set UpdateAction to uaApplied before exiting.

**See Also**

- [CachedUpdates](#)

© 1997-2013 Devart. All Rights Reserved.

**17.26.2 Variables**

Variables in the **MemDS** unit.

**Variables**

Name	Description
<a href="#">DoNotRaiseExcetionOnUaFail</a>	An exception will be raised if the value of the UpdateAction parameter is uaFail.
<a href="#">SendDataSetChangeEventAfterOpen</a>	The DataSetChangeEvent is sent after a dataset gets open. It was necessary to fix a problem with disappeared vertical scrollbar in some types of DB-aware grids.

© 1997-2013 Devart. All Rights Reserved.

**17.26.2.1 DoNotRaiseExcetionOnUaFail Variable**

An exception will be raised if the value of the UpdateAction parameter is uaFail.

**Unit**

[MemDS](#)**Syntax**

```
DoNotRaiseExcetionOnUaFail: boolean = False;
```

**Remarks**

Starting with IBDAC 2.20.0.12, if the [OnUpdateRecord](#) event handler sets the UpdateAction parameter to uaFail, an exception is raised. The default value of UpdateAction is uaFail. So, the exception will be raised when the value of this parameter is left unchanged.

To restore the old behaviour, set DoNotRaiseExcetionOnUaFail to True.

© 1997-2013 Devart. All Rights Reserved.

**17.26.2.2 SendDataSetChangeEventAfterOpen Variable**

The DataSetChange event is sent after a dataset gets open. It was necessary to fix a problem with disappeared vertical scrollbar in some types of DB-aware grids.

**Unit**[MemDS](#)**Syntax**

```
SendDataSetChangeEventAfterOpen: boolean = True;
```

**Remarks**

Starting with IBDAC 2.20.0.11, the DataSetChange event is sent after a dataset gets open. It was necessary to fix a problem with disappeared vertical scrollbar in some types of DB-aware grids. This problem appears only under Windows XP when visual styles are enabled.

To disable sending this event, change the value of this variable to False.

© 1997-2013 Devart. All Rights Reserved.

**17.27 MemUtils**

This unit contains auxiliary procedures and functions used in the DAC code.

**Routines**

Name	Description
<a href="#">Reverse4</a>	Switches places of bytes in the cardinal argument.

© 1997-2013 Devart. All Rights Reserved.

**17.27.1 Routines**

Routines in the **MemUtils** unit.

**Routines**

Name	Description
------	-------------

[Reverse4](#)

Switches places of bytes in the cardinal argument.

© 1997-2013 Devart. All Rights Reserved.

### 17.27.1.1 Reverse4 Function

Switches places of bytes in the cardinal argument.

#### Unit

[MemUtils](#)

#### Syntax

```
function Reverse4(Value: cardinal): cardinal;
```

#### Parameters

*Value*

Holds the input value.

#### Return Value

the output value.

© 1997-2013 Devart. All Rights Reserved.

## 17.28 VirtualTable

This unit contains implementation of the TVirtualTable component.

#### Classes

Name	Description
<a href="#">TVirtualTable</a>	A base class for storing data in memory.

#### Types

Name	Description
<a href="#">TVirtualTableOptions</a>	Represents the set of <a href="#">TVirtualTableOption</a> .

#### Enumerations

Name	Description
<a href="#">TVirtualTableOption</a>	Specifies the actions to take on fields data at the time of opening or closing TVirtualTable dataset.

© 1997-2013 Devart. All Rights Reserved.

## 17.28.1 Classes

Classes in the **VirtualTable** unit.

### Classes

Name	Description
<a href="#">TVirtualTable</a>	A base class for storing data in memory.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.28.1.1 TVirtualTable Class

A base class for storing data in memory.

For a list of all members of this type, see [TVirtualTable](#) members.

### Unit

[VirtualTable](#)

### Syntax

```
TVirtualTable = class (TMemDataSet) ;
```

### Remarks

TVirtualTable is inherited from the TMemDataSet component. TVirtualTable stores data in memory and does not have linked data files. To add fields to virtual table at design time use Fields Editor. Call the [TVirtualTable.AddField](#) method to add fields at run time.

Immediately after creating, virtual table will be empty. Then you define new fields or load existing table files so that the virtual table object becomes initialized and ready to be opened.

When you close virtual table it will discard its record set. To keep the data you entered at design-time for later use you may wish to include the `voStored` option in the [TVirtualTable.Options](#) property. At run time you will need to call the [TVirtualTable.SaveToFile](#) method explicitly to store modifications to the file that may be retrieved back into the virtual table by calling the [TVirtualTable.LoadFromFile](#) method later.

**Note:** TVirtualTable component is added to the Data Access page of the component palette, not to the InterBase Access page.

### Inheritance Hierarchy

[TMemDataSet](#)

**TVirtualTable**

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.28.1.1.1 Members

[TVirtualTable](#) class overview.

### Properties

Name	Description
------	-------------

---

<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">Options</a>	Used to specify actions to take on the fields data at the time of opening or closing TVirtualTable dataset.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.

## Methods

Name	Description
<a href="#">AddField</a>	Adds a new TFieldDef object with the name determined by Name.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">Assign</a>	Copies fields and data from another TDataSet component.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">Clear</a>	Removes all records from TVirtualTable.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.

[DeleteField](#)

Deletes a field specified by name.

[DeleteFields](#)

Deletes all fields.

[GetBlob](#) (inherited from [TMemDataSet](#))

Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.

[LoadFromFile](#)

Loads data from file into a TVirtualTable component.

[LoadFromStream](#)

Copies data of a stream into a TVirtualTable component.

[Locate](#) (inherited from [TMemDataSet](#))

Overloaded. Searches a dataset for a specific record and positions the cursor on it.

[LocateEx](#) (inherited from [TMemDataSet](#))

Overloaded. Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet.

[Prepare](#) (inherited from [TMemDataSet](#))

Allocates resources and creates field components for a dataset.

[RestoreUpdates](#) (inherited from [TMemDataSet](#))

Marks all records in the cache of updates as unapplied.

[RevertRecord](#) (inherited from [TMemDataSet](#))

Cancels changes made to the current record when cached updates are enabled.

[SaveToFile](#)

Saves data of a TVirtualTable component to a file.

[SaveToStream](#)

Copies data from a TVirtualTable component to a stream.

[SaveToXML](#) (inherited from [TMemDataSet](#))

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

[UnPrepare](#) (inherited from [TMemDataSet](#))

Frees the resources allocated for a previously prepared query on the server and client sides.

[UpdateResult](#) (inherited from [TMemDataSet](#))

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Indicates the current update status for the dataset when cached updates are enabled.

## Events

Name	Description
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.

© 1997-2013 Devart. All Rights Reserved.

### 17.28.1.1.2 Properties

Properties of the **TVirtualTable** class.

For a complete list of the **TVirtualTable** class members, see the [TVirtualTable Members](#) topic.

## Public

Name	Description
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">GetBlob</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.
<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.
<a href="#">Prepare</a> (inherited from <a href="#">TMemDataSet</a> )	Allocates resources and creates field components for a dataset.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the <a href="#">ApplyUpdates</a> method while cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.



[UpdateStatus](#) (inherited from [TMemDataSet](#))

Indicates the current update status for the dataset when cached updates are enabled.

## Published

Name	Description
<a href="#">Options</a>	Used to specify actions to take on the fields data at the time of opening or closing TVirtualTable dataset.

## See Also

- [TVirtualTable Class](#)
- [TVirtualTable Class Members](#)

© 1997-2013 Devart. All Rights Reserved.

### 17.28.1.1.2.1 Options Property

Used to specify actions to take on the fields data at the time of opening or closing TVirtualTable dataset.

## Class

[TVirtualTable](#)

## Syntax

```
property Options: TVirtualTableOptions default [voPersistentData, voStored];
```

## Remarks

The Options property specifies what actions to take on the fields data at the time of opening or closing TVirtualTable dataset.

© 1997-2013 Devart. All Rights Reserved.

### 17.28.1.1.3 Methods

Methods of the **TVirtualTable** class.

For a complete list of the **TVirtualTable** class members, see the [TVirtualTable Members](#) topic.

## Public

Name	Description
<a href="#">AddField</a>	Adds a new TFieldDef object with the name determined by Name.
<a href="#">ApplyUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Writes dataset's pending cached updates to a database.

<a href="#">Assign</a>	Copies fields and data from another TDataSet component.
<a href="#">CachedUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Used to enable or disable the use of cached updates for a dataset.
<a href="#">CancelUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears all pending cached updates from cache and restores dataset in its prior state.
<a href="#">Clear</a>	Removes all records from TVirtualTable.
<a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Clears the cached updates buffer.
<a href="#">DeferredPost</a> (inherited from <a href="#">TMemDataSet</a> )	Makes permanent changes to the database server.
<a href="#">DeleteField</a>	Deletes a field specified by name.
<a href="#">DeleteFields</a>	Deletes all fields.
<a href="#">GetBlob</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
<a href="#">IndexFieldNames</a> (inherited from <a href="#">TMemDataSet</a> )	Used to get or set the list of fields on which the recordset is sorted.
<a href="#">LoadFromFile</a>	Loads data from file into a TVirtualTable component.
<a href="#">LoadFromStream</a>	Copies data of a stream into a TVirtualTable component.
<a href="#">LocalConstraints</a> (inherited from <a href="#">TMemDataSet</a> )	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
<a href="#">LocalUpdate</a> (inherited from <a href="#">TMemDataSet</a> )	Used to prevent implicit update of rows on database server.
<a href="#">Locate</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
<a href="#">LocateEx</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Excludes features that don't need to be included to the <a href="#">TMemDataSet.Locate</a> method of TDataSet.

<a href="#">OnUpdateError</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when an exception is generated while cached updates are applied to a database.
<a href="#">OnUpdateRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Occurs when a single update component can not handle the updates.
<a href="#">Prepare</a> (inherited from <a href="#">TMemDataSet</a> )	Allocates resources and creates field components for a dataset.
<a href="#">Prepared</a> (inherited from <a href="#">TMemDataSet</a> )	Determines whether a query is prepared for execution or not.
<a href="#">RestoreUpdates</a> (inherited from <a href="#">TMemDataSet</a> )	Marks all records in the cache of updates as unapplied.
<a href="#">RevertRecord</a> (inherited from <a href="#">TMemDataSet</a> )	Cancels changes made to the current record when cached updates are enabled.
<a href="#">SaveToFile</a>	Saves data of a TVirtualTable component to a file.
<a href="#">SaveToStream</a>	Copies data from a TVirtualTable component to a stream.
<a href="#">SaveToXML</a> (inherited from <a href="#">TMemDataSet</a> )	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
<a href="#">UnPrepare</a> (inherited from <a href="#">TMemDataSet</a> )	Frees the resources allocated for a previously prepared query on the server and client sides.
<a href="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )	Used to indicate the update status for the current record when cached updates are enabled.
<a href="#">UpdateResult</a> (inherited from <a href="#">TMemDataSet</a> )	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
<a href="#">UpdatesPending</a> (inherited from <a href="#">TMemDataSet</a> )	Used to check the status of the cached updates buffer.
<a href="#">UpdateStatus</a> (inherited from <a href="#">TMemDataSet</a> )	Indicates the current update status for the dataset when cached updates are enabled.

**See Also**

- [TVirtualTable Class](#)

- [TVirtualTable Class Members](#)

---

© 1997-2013 Devart. All Rights Reserved.

#### 17.28.1.1.3.1 AddField Method

Adds a new TFieldDef object with the name determined by Name.

### Class

[TVirtualTable](#)

### Syntax

```
procedure AddField(Name: string; FieldType: TFieldType; Size: integer = 0; Required: boolean = False);
```

### Parameters

#### *Name*

Holds the name of the TFieldDef object to add.

#### *FieldType*

Holds the type of the TFieldDef object to add.

#### *Size*

Holds the size of the string (if the type of TFieldDef object was specified as ftString or ftWideString).

#### *Required*

Holds an indicator that determines whether filling the Size parameter is required.

### Remarks

Call the AddField method to add a new TFieldDef object with the name determined by Name. FieldType can be ftString, ftWideString, ftSmallint, ftInteger, ftAutoInc, ftWord, ftBoolean, ftLargeint, ftFloat, ftCurrency, ftDate, ftTime, ftDateTime, ftBlob, or ftMemo. When you add ftString or ftWideString field you should specify Size of the string.

### Example

```
VirtualTable1.AddField('CODE', ftInteger, 0);  
VirtualTable1.AddField('NAME', ftString, 30);
```

### See Also

- [DeleteField](#)
  - [DeleteFields](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.28.1.1.3.2 Assign Method

Copies fields and data from another TDataSet component.

**Class**

[TVirtualTable](#)

**Syntax**

```
procedure Assign(Source: TPersistent); override;
```

**Parameters**

*Source*

Holds the TDataSet component to copy fields and data from.

**Remarks**

Call the Assign method to copy fields and data from another TDataSet component.

**Note:** Unsupported field types are skipped (i.e. destination dataset will contain less fields than the source one). This may happen when Source is not a TVirtualTable component but some SQL server oriented dataset.

**Example**

```
IBCQuery1.SQL.Text := 'SELECT * FROM DEPT';  
IBCQuery1.Active := True;  
VirtualTable1.Assign(IBCQuery1);  
VirtualTable1.Active := True;
```

**See Also**

- 

[TVirtualTable](#)

---

© 1997-2013 Devart. All Rights Reserved.

## 17.28.1.1.3.3 Clear Method

Removes all records from TVirtualTable.

**Class**

[TVirtualTable](#)

**Syntax**

```
procedure Clear;
```

**Remarks**

Call the Clear method to remove all records from TVirtualTable.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.28.1.1.3.4 DeleteField Method

Deletes a field specified by name.

**Class**

[TVirtualTable](#)

**Syntax**

```
procedure DeleteField(Name: string);
```

**Parameters**

*Name*

Holds the name of the field to delete.

**Remarks**

Call the DeleteField method to delete a field specified by Name.

**See Also**

- [AddField](#)
  - [DeleteFields](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.28.1.1.3.5 DeleteFields Method

Deletes all fields.

**Class**

[TVirtualTable](#)

**Syntax**

```
procedure DeleteFields;
```

**Remarks**

Call the DeleteFields method to delete all fields.

**See Also**

- [DeleteField](#)
- 

© 1997-2013 Devart. All Rights Reserved.

## 17.28.1.1.3.6 LoadFromFile Method

Loads data from file into a TVirtualTable component.

**Class**

[TVirtualTable](#)

## Syntax

```
procedure LoadFromFile(const FileName: string; LoadFields: boolean  
= True);
```

### Parameters

#### *FileName*

Holds the name of the file to load data from.

#### *LoadFields*

Indicates whether to load fields from the file.

## Remarks

Call the LoadFromFile method to load data from file into a TVirtualTable component. Specify the name of the file to load into the field as the value of the FileName parameter. This file may be an XML document in ADO-compatible format or in virtual table data format. File format will be detected automatically.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.28.1.1.3.7 LoadFromStream Method

Copies data of a stream into a TVirtualTable component.

## Class

[TVirtualTable](#)

## Syntax

```
procedure LoadFromStream(Stream: TStream; LoadFields: boolean =  
True);
```

### Parameters

#### *Stream*

Holds the stream from which the field's value is copied.

#### *LoadFields*

Indicates whether to load fields from the stream.

## Remarks

Call the LoadFromStream method to copy data of a stream into a TVirtualTable component. Specify the stream from which the field's value is copied as the value of the Stream parameter. Data in the stream may be in ADO-compatible format or in virtual table data format. Data format will be detected automatically.

---

© 1997-2013 Devart. All Rights Reserved.

### 17.28.1.1.3.8 SaveToFile Method

Saves data of a TVirtualTable component to a file.

## Class

[TVirtualTable](#)

## Syntax

```
procedure SaveToFile(const FileName: string; StoreFields: boolean
= True);
```

**Parameters***FileName*

Holds the name of the file to save data to.

*StoreFields*

Indicates whether to save fields to a file.

**Remarks**

Call the SaveToFile method to save data of a TVirtualTable component to a file. Specify the name of the file as the value of the FileName parameter.

---

© 1997-2013 Devart. All Rights Reserved.

## 17.28.1.1.3.9 SaveToStream Method

Copies data from a TVirtualTable component to a stream.

**Class**

[TVirtualTable](#)

**Syntax**

```
procedure SaveToStream(Stream: TStream; StoreFields: boolean =
True);
```

**Parameters***Stream*

Holds the name of the stream to which the field's value is saved.

*StoreFields*

Indicates whether to save the fields names to a file.

**Remarks**

Call the SaveToStream method to copy data from a TVirtualTable component to a stream. Specify the name of the stream to which the field's value is saved as the value of the Stream parameter.

---

© 1997-2013 Devart. All Rights Reserved.

**17.28.2 Types**

Types in the **VirtualTable** unit.

**Types**

Name	Description
<a href="#">TVirtualTableOptions</a>	Represents the set of <a href="#">TVirtualTableOption</a> .

---

© 1997-2013 Devart. All Rights Reserved.



### 17.28.2.1 TVirtualTableOptions Set

Represents the set of [TVirtualTableOption](#).

#### Unit

[VirtualTable](#)

#### Syntax

```
TVirtualTableOptions = set of TVirtualTableOption;
```

© 1997-2013 Devart. All Rights Reserved.

### 17.28.3 Enumerations

Enumerations in the **VirtualTable** unit.

#### Enumerations

Name	Description
<a href="#">TVirtualTableOption</a>	Specifies the actions to take on fields data at the time of opening or closing TVirtualTable dataset.

© 1997-2013 Devart. All Rights Reserved.

#### 17.28.3.1 TVirtualTableOption Enumeration

Specifies the actions to take on fields data at the time of opening or closing TVirtualTable dataset.

#### Unit

[VirtualTable](#)

#### Syntax

```
TVirtualTableOption = (voPersistentData, voStored);
```

#### Values

Value	Meaning
<b>voPersistentData</b>	Dataset will not dispose of its data at the time of dataset closing.
<b>voStored</b>	Dataset will keep its data set at design-time in DFM file along with other form's stored properties.

© 1997-2013 Devart. All Rights Reserved.

# Index

## - 6 -

64-bit Development with Embarcadero RAD Studio XE2 59

## - A -

AbortOnKeyViol Property 73  
 AbortOnProblem Property 74  
 Action Property  
     TIBCLicensingService 650  
     TIBCLimboTransactionInfo 654  
 ActivateShadow Method 623  
 Active Property  
     TCustomDASQLMonitor 160  
     TCustomIBCSERVICE 602  
     TDAAlerter 98  
     TDATransaction 338  
     TIBCTransaction 580  
     TMacro 343  
 ActiveUser Property 693  
 AddAlias Method 678  
 AddConnection Method 584  
 AddDBTypeRule Method 303  
 AddDrop Property 115  
 AddField Method 880  
 AddFieldNameRule Method 309  
 AddLicense Method 652  
 AddRef Method 840  
 AddRule Method 311  
 AddTransaction Method 465  
 AddUser Method 669  
 AddWhere Method 235  
 Advise Property 655  
 AfterExecute Event  
     TCustomDADataset 253  
     TCustomDASQL 271  
     TDAScript 147  
 AfterFetch Event 253  
 AfterUpdateExecute Event 253  
 AliasCount Property 675  
 Allocation Property 760  
 AllocBlob Method 785

AlterJournal Method 624  
 Apply Method 279  
 ApplyUpdates Method  
     ApplyUpdates 855, 856  
     TCustomDAConnection 191  
     TMemDataSet 855  
 ARRAY Data Type 49  
 ArrayDimensions Property 726  
 ArrayHighBound Property(Indexer) 726  
 ArrayID Property 727  
 ArrayLowBound Property(Indexer) 727  
 ArraySize Property 728  
 AsArray Property  
     TIBCArryField 445  
     TIBCParm 497  
 AsBlob Property 325  
 AsBlobRef Property 325  
 AsDateTime Property 344  
 AsFloat Property  
     TDAParm 326  
     TMacro 344  
 AslBlob Property 497  
 AsInteger Property  
     TDAParm 326  
     TMacro 344  
 AsLargeInt Property 326  
 AsMemo Property 327  
 AsMemoRef Property 327  
 Assign Method  
     TBlob 824  
     TCustomIBCArray 735  
     TVirtualTable 881  
 AssignConnect Method 466  
 AssignField Method 331  
 AssignFieldValue Method 331  
 AssignValues Method 348  
 AsSQLTimeStamp Property 327  
 AsString Property  
     TBlob 821  
     TCustomIBCArray 728  
     TDAParm 328  
     TMacro 345  
 AsWideString Property  
     TBlob 822  
     TDAParm 328  
 Attach Method 604  
 AttachmentID Property 760  
 AttributeByName Method 837

AttributeCount Property 835  
 AttributeNo Property 816  
 Attributes Property(Indexer) 835  
 AutoClose Property 481  
 AutoCommit Property  
     TCustomIBCDDataSet 386  
     TIBCAlerter 717  
     TIBCCConnection 452  
 AutoDDL Property 804  
 Automatic Key Field Value Generation 39  
 AutoPrepare Property 288  
 AutoRegister Property 99

## - B -

BackoutCount Property 761  
 Backup Method 108  
 BackupFile Property  
     TIBCBackupRestoreService 608  
     TIBCBackupService 613  
     TIBCRestoreService 661  
 BackupQuery Method 109  
 BackupToFile Method 109  
 BackupToStream Method 110  
 BaseLevel Property 761  
 BaseLocation Property 616  
 BaseSQL Property 215  
 BeforeExecute Event 147  
 BeforeFetch Event 254  
 BeforeUpdateExecute Event 254  
 BLOB Data Types 46  
 BlockingFactor Property 613  
 bmAppend 81  
 bmAppendUpdate 81  
 bmDelete 81  
 bmUpdate 81  
 boConvertExtTables 709  
 bolgnoreChecksums 709  
 bolgnoreLimbo 709  
 boMetadataOnly 709  
 boNoGarbageCollection 709  
 boNonTransportable 709  
 boOldMetadataDesc 709  
 BooleanDomainFields Property 481  
 BreakExec Method  
     TCustomDADataSet 236  
     TDAScript 142  
     TIBCSQL 530

BringDatabaseOnline Method 624  
 BufferSize Property 635

## - C -

CacheArrays Property 481  
 CacheBlobs Property 482  
 CacheCalcFields Property 288  
 Cached Property  
     TCustomIBCArray 728  
     TIBCBlob 779  
 CachedDimensions Property 729  
 CachedHighBound Property(Indexer) 729  
 CachedLowBound Property(Indexer) 729  
 CachedSize Property 730  
 CachedUpdates Property 850  
 CancelButton Property 176  
 CancelUpdates Method 856  
 CapabilityMask Property 647  
 Caption Property 177  
 cbClient 842  
 cbClientServer 842  
 cbNone 842  
 cbServer 842  
 ChangeCursor Property 258  
 ChangeCursor Variable 359  
 ChangedCount Property 74  
 CharLength Property 472  
 Charset Property 472  
 CharsetId Property 779  
 CheckpointInterval Property 640  
 CheckpointLength Property 641  
 clApply 843  
 clConnect 843  
 clConnectionApply 843  
 Clear Method  
     TBlob 824  
     TVirtualTable 881  
 ClearArray Method 735  
 clExecute 843  
 ClientLibrary Property 454  
 clOpen 843  
 CloseBlob Method 786  
 clRefresh 843  
 clServiceQuery 843  
 clTransStart 843  
 clUnknown 843  
 ColumnName Property 730

- Columns Property
  - TDALoader 124
  - TIBCLoader 797
- Commit Method
  - TCustomDAConnection 192
  - TDATransaction 339
  - TIBCTransaction 585
- CommitCount Property 74
- CommitRetaining Method
  - TIBCCConnection 466
  - TIBCTransaction 585
- CommitUpdates Method 857
- Compatibility 23
- Compatibility with Previous Versions 58
- ComplexArrayFields Property 482
- Component List 18
- Component Property 173
- CompressBlobMode Property 288
- Compressed Property 832
- CompressedSize Property 832
- Config Property 688
- ConfigParams Property 675
- Connect Method 192
- ConnectButton Property 177
- ConnectDialog Property 184
- Connected Property 454
- Connection Pooling 55
- Connection Property
  - TCustomDADataset 215
  - TCustomDASQL 258
  - TCustomIBCDataset 386
  - TDAAlerter 99
  - TDADump 105
  - TDALoader 124
  - TDAMetaData 317
  - TDAScript 137
  - TIBCAlerter 717
  - TIBCCConfigService 620
  - TIBCCConnectDialog 811
  - TIBCScript 805
  - TIBCSQL 525
- ConnectionLifetime Property 352
- Connections Property(Indexer) 580
- Connections Variable 595
- ConnectionsCount Property 581
- ConnectPrompt Property 454
- ConversionCharsetID Property 780
- ConversionSubType Property 780

- ConvertEOL Property 185
- CRAccess Unit Members 66
- CRBatchMove Unit Members 69
- CRDataTypeMap Unit Members 82
- CreateBlobStream Method 236
- CreateColumns Method 125
- CreateDatabase Method 467
- CreateDataSet Method 193
- CreateJournal Method 624
- CreateJournalArchive Method 625
- CreateProcCall Method
  - TCustomIBCDataset 401
  - TIBCSQL 530
- CreateSQL Method 193
- CreateTemporaryArray Method 736
- CREncryption Unit Members 89
- CurLogFileName Property 761
- CurLogPartitionOffset Property 762
- CurrentMemory Property 762
- Cursor Property 386

## - D -

- DAAlterter Unit Members 96
- DACProductName Constant 752
- DADDataAdapter Class 360
- DADDataAdapter.DataSet Property 362
- DADDataAdapter.Fill Method 363
- DADDataAdapter.Update Method 363
- DADump Unit Members 102
- DALoader Unit Members 117
- DAScript Unit Members 132
- DASQLMonitor Unit Members 158
- Data Encryption 44
- Data Type Mapping 40
- Database Property
  - TIBCBBackupService 614
  - TIBCCConfigService 621
  - TIBCCConnection 455
  - TIBCRestoreService 662
  - TIBCStatisticalService 684
  - TIBCValidationService 700
- Database Specific Aspects of 64-bit Development 63
- DatabasInfo Property
  - TIBCCConnection 456
  - TIBCServerProperties 675
- DatabaseLabel Property 812

- DataHeader Property 91
- DataSet Property
  - DADDataAdapter 362
  - TCustomDAUpdateSQL 274
  - TDAScript 137
  - TIBCScript 805
- DataSize Property 816
- DataType Property
  - TAttribute 817
  - TDAParam 328
  - TObjectType 836
- DBAccess Unit Members 168
- DBFileName Property 762
- DbHandle Property
  - TCustomIBCArrary 731
  - TIBCBlob 781
- DBImplementationClass Property 763
- DBImplementationNo Property 763
- DBLengthMax Property
  - TDAMapRule 299
  - TMapRule 87
- DBLengthMin Property
  - TDAMapRule 299
  - TMapRule 87
- DBMonitorOptions Property 160
- DbName Property 638
- DBScaleMax Property
  - TDAMapRule 299
  - TMapRule 87
- DBScaleMin Property
  - TDAMapRule 300
  - TMapRule 87
- DBSiteName Property 763
- DBSQLDialect Property 456
- DBType Property
  - TDAMapRule 300
  - TMapRule 88
- Debug Property
  - TCustomDADataset 216
  - TCustomDASQL 259
  - TDADump 106
  - TDAScript 138
  - TIBCCConnection 456
- DefaultCloseAction Property
  - TDATransaction 339
  - TIBCTransaction 581
- DefaultConnection Property 582
- DefaultRole Property 694
- DefaultSortType Property 282
- DefaultTransaction Property 457
- DefaultValues Property
  - TDADatasetOptions 289
  - TIBCDatasetOptions 483
- DefConnection Variable 595
- DeferredArrayRead Property 483
- DeferredBlobRead Property 483
- DeferredPost Method 858
- DeleteAlias Method 678
- DeleteCount Property 763
- DeleteField Method 882
- DeleteFields Method 882
- DeleteObject Property 274
- DeleteSQL Property 274
- DeleteTable Method 437
- DeleteUser Method 670
- DeleteWhere Method 236
- Delimiter Property 138
- Demo Projects 13
- Deployment 28
- Desc Property 644
- DescribeParams Property
  - TIBCDatasetOptions 484
  - TIBCSQL 526
- Description Property 694
- Destination Property 75
- Detach Method 604
- DetailDelay Property 289
- DetailFields Property 216
- Devart.Dac.DataAdapter Unit Members 360
- Devart.IbDac.DataAdapter Unit Members 364
- DialogClass Property 177
- Directory Property 641
- DisableFlush Method 625
- Disconnect Method 194
- Disconnected Mode 52
- Disconnected Property 217
- DisconnectedMode Property 282
- DisplayUser Method 670
- DisplayUsers Method 670
- DMLRefresh Property 387
- DoNotRaiseExcetionOnUaFail Variable 869
- DropDatabase Method 467
- DropJournal Method 625
- DropJournalArchive Method 626

## - E -

eaAbort 157  
 eaAES128 95  
 eaAES192 95  
 eaAES256 95  
 eaBlowfish 95  
 eaCast128 95  
 eaContinue 157  
 eaException 157  
 eaFail 157  
 eaRC4 95  
 eaTripleDES 95  
 EDAError Class 172  
 EDAError.Component Property 173  
 EDAError.ErrorCode Property 173  
 EDataMappingError Class 83  
 EDataTypeMappingError Class 83  
 Editions 22  
 ehNone 94  
 ehTag 94  
 ehTagAndHash 94  
 EIBCErrors Class 792  
 EIBCErrors.ErrorNumber Property 794  
 EIBCErrors.Sender Property 794  
 EIBCErrors.SQLErrorMsg Property 794  
 EInvalidDBTypeMapping Class 84  
 EInvalidFieldTypeMapping Class 84  
 EmptyTable Method 437  
 EnableBCD Property 473  
 EnableEUA Method 671  
 EnableFMTBCD Property 473  
 EnableMemos Property 473  
 Encryption Property 217  
 EncryptionAlgorithm Property 91  
 Encryptor Property 296  
 EndLine Property  
     TDAScript 138  
     TDAScript 150  
 EndOffset Property  
     TDAScript 139  
     TDAScript 150  
 EndPos Property  
     TDAScript 139  
     TDAScript 151  
 Eof Property 635  
 ErrorCode Property 173

ErrorNumber Property 794  
 ErrorOffset Method 143  
 EUnsupportedDataTypeMapping Class 85  
 Events Property 717  
 ExecProc Method  
     TCustomDAConnection 194  
     TIBCStoredProc 557  
 ExecProcEx Method 196  
 ExecSQL Method  
     TCustomDAConnection 197  
     TCustomDAUpdateSQL 279  
 ExecSQLEx Method 198  
 Execute Method  
     Execute 265, 266  
     TCRBatchMove 79  
     TCustomConnectDialog 180  
     TCustomDADataset 237  
     TCustomDASQL 265  
     TDAScript 143  
     TDAScript 153  
 ExecuteFile Method 144  
 ExecuteNext Method  
     TDAScript 144  
     TIBCSQL 531  
 ExecuteStream Method 145  
 Executing Method  
     TCustomDADataset 237  
     TCustomDASQL 266  
 Exists Property 429  
 ExpungeCount Property 764

## - F -

Features 7  
 Fetch Method 679  
 FetchAliasInfo Method 679  
 FetchAll Property  
     TIBCQuery 519  
     TIBCTable 575  
 FetchConfigParams Method 679  
 FetchDatabaseInfo Method 680  
 Fetched Method  
     TCustomDADataset 238  
     TCustomIBCDataset 401  
 Fetches Property 764  
 Fetching Method 238  
 FetchingAll Method 239  
 FetchLicenseInfo Method 680

FetchLicenseMaskInfo Method 680  
FetchLimboTransactionInfo Method 704  
FetchRows Property 217  
FetchVersionInfo Method 681  
FieldLength Property  
    TDAMapRule 300  
    TMapRule 88  
FieldMappingMode Property 75  
FieldName Property  
    TDAMapRule 301  
    TMapRule 88  
Fields Property 296  
FieldsAsString Property 484  
FieldScale Property  
    TDAMapRule 301  
    TMapRule 88  
FieldsOrigin Property 289  
FieldType Property  
    TDAColumn 119  
    TDAMapRule 301  
Fill Method 363  
FilterSQL Property 218  
FinalSQL Property  
    TCustomDADataset 218  
    TCustomDASQL 259  
FindAttribute Method 838  
FindDefaultConnection Method 585  
FindDefaultTransaction Method 467  
FindKey Method 239  
FindMacro Method  
    TCustomDADataset 240  
    TCustomDASQL 266  
    TDAScript 145  
    TMacros 348  
FindNearest Method 240  
FindParam Method  
    TCustomDADataset 241  
    TCustomDASQL 267  
    TCustomIBCDataset 401  
    TDAParams 336  
    TIBCPParams 502  
    TIBCSQL 531  
FirstName Property 694  
FixLimboTransactionErrors Method 704  
FlatBuffers Property 290  
FlushDatabase Method 626  
ForcedWrites Property 764  
FreeBlob Method 786

Frequently Asked Questions 31  
FullRefresh Property 484

## - G -

GDSDLL Variable 751  
GDSVersion Variable 752  
GDSVersionSt Variable 752  
GenerateHeader Property 116  
GeneratorMode Property 388  
GeneratorStep Property 388  
GetArray Method 402  
GetArrayInfo Method 736  
GetBlob Method  
    TCustomIBCDataset 403  
    TMemDataSet 858  
GetDatabaseNames Method 199  
GetDataType Method 241  
GetFieldObject Method 242  
GetFieldPrecision Method 242  
GetFieldScale Method 243  
GetItemAsDateTime Method 737  
GetItemAsFloat Method 737  
GetItemAsInteger Method 738  
GetItemAsSmallInt Method 738  
GetItemAsString Method 739  
GetItemAsWideString Method 740  
GetItemsSlice Method 740  
GetItemValue Method 741  
GetJournalInformation Method 626  
GetMetaDataKinds Method 321  
GetNextChunk Method 636  
GetNextLine Method 637  
GetOrderBy Method 244  
GetRestrictions Method 321  
GetServerList Method 181  
GetStoredProcNames Method 199  
GetTableNames Method 200  
Getting Started 3  
Getting Support 30  
GlobalAction Property 701  
gmInsert 592  
gmPost 592  
GotoCurrent Method 244  
GroupID Property 694  
GroupName Property 695



## - H -

haMD5 95  
 Handle Property  
     TCustomIBCDataSet 388  
     TCustomIBCSERVICE 602  
     TIBCBlob 781  
     TIBCCONNECTION 457  
     TIBCSQL 526  
     TIBCTransaction 582  
 HasArchive Property 641  
 haSHA1 95  
 HashAlgorithm Property 92  
 HasJournal Property 642  
 Hierarchy Chart 20  
 Host Property 164  
 HostSite Property 655

## - I -

IBC Unit Members 366  
 IBCAdmin Unit Members 596  
 IBCAlerter Unit Members 714  
 IBCArray Unit Members 721  
 IBCCall Unit Members 748  
 IBCClasses Unit Members 753  
 IBCCONNECTIONPool Unit Members 791  
 IBCDataAdapter Class 365  
 IBCErrors Unit Members 792  
 IBCLoader Unit Members 795  
 IBScript Unit Members 800  
 IBSQLMonitor Unit Members 806  
 IbDacVcl Unit Members 808  
 IBDACVersion Constant 596  
 iblCustom 788  
 iblReadCommitted 788  
 iblReadOnlyReadCommitted 788  
 iblReadOnlyTableStability 788  
 iblSnapshot 788  
 iblTableStability 788  
 IBXMLDLL Variable 752  
 ID Property  
     TIBCBlob 782  
     TIBCLicenseInfo 645  
     TIBCLicensingService 650  
     TIBCLimboTransactionInfo 655

IgnoreErrors Property  
     TDAMapRule 302  
     TMapRule 89  
 ihFail 96  
 ihIgnoreError 96  
 ihSkipData 96  
 ilReadCommitted 68  
 Increasing Performance 53  
 IndexFieldNames Property 851  
 InsertCount Property 765  
 InsertObject Property 275  
 InsertSQL Property 275  
 Installation 25  
 IntegerPrecision Variable 790  
 InTransaction Property 185  
 InvalidHashAction Property 92  
 IsEqual Method 349  
 IsInit Method 786  
 IsNull Property  
     TCustomIBCArray 731  
     TDAParam 329  
 IsolationLevel Property 582  
 IsQuery Property  
     TCustomDADataset 219  
     TCustomIBCDataSet 389  
 IsRemoteConnect Property 765  
 IsUnicode Property 822  
 Items Property 731  
 Items Property(Indexer)  
     TDAColumns 121  
     TDAParams 335  
     TDAStatements 155  
     TIBCPARAMS 501  
     TMacros 347  
 ItemScale Property 732  
 ItemSize Property 732  
 ItemType Property 443

## - J -

JournalInformation Property 621

## - K -

KeepDesignConnected Property 282  
 Key Property  
     TIBCLicenseInfo 645



Key Property  
     TIBCLicensingService 651  
 KeyFields Property 219  
 KeyGenerator Property 389  
 KeyViolCount Property 75

## - L -

laAdd 710  
 LabelSet Property 178  
 laRemove 710  
 LastError Property 458  
 LastName Property 695  
 Length Property 817  
 LengthBlob Method 787  
 LicensedUsers Property 645  
 LicenseInfo Property 676  
 LicenseMask Property 647  
 LicenseMaskInfo Property 676  
 Licensing and Subscriptions 29  
 LimboTransactionInfo Property(Indexer) 701  
 LimboTransactionInfoCount Property 701  
 ListTraceSessions Method 690  
 ImLockDelayed 358  
 ImLockImmediate 358  
 ImNone 358  
 Load Method 126  
 LoadFromDataSet Method 126  
 LoadFromFile Method  
     TBlob 825  
     TDAParam 332  
     TVirtualTable 882  
 LoadFromStream Method  
     TBlob 825  
     TDAParam 332  
     TVirtualTable 883  
 LocalConstraints Property 851  
 LocalFailover Property 283  
 LocalMasterDetail Property 290  
 LocalUpdate Property 852  
 Locate Method 859  
 LocateEx Method 861  
 Lock Method 245  
 LockFileLocation Property 616  
 LockMode Property  
     TCustomIBCDataSet 390  
     TIBCQuery 520  
     TIBCStoredProc 549

TIBCTable 576  
 LockObject Property 276  
 LockSQL Property 276  
 LogFile Property 765  
 LoginPrompt Property  
     TCustomDAConnection 185  
     TCustomIBCSERVICE 602  
 LongStrings Property 290  
 IsCustom 357  
 IsEnglish 357  
 IsFrench 357  
 IsGerman 357  
 IsItalian 357  
 IsPolish 357  
 IsPortuguese 357  
 IsRussian 357  
 IsSpanish 357  
 IxCasInsensitive 844  
 IxNearest 844  
 IxNext 844  
 IxPartialCompare 844  
 IxPartialKey 844  
 IxUp 844

## - M -

MacroByName Method  
     TCustomDADataset 245  
     TCustomDASQL 267  
     TDAScript 146  
     TMacros 349  
 MacroChar Variable 360  
 MacroCount Property  
     TCustomDADataset 220  
     TCustomDASQL 260  
 Macros 56  
 Macros Property  
     TCustomDADataset 220  
     TCustomDASQL 260  
     TDAScript 139  
 Mappings Property 76  
 Marks Property 766  
 Master/Detail Relationships 37  
 MasterFields Property 221  
 MasterSource Property 221  
 MaxMemory Property 766  
 MaxPoolSize Property 352  
 MaxSegmentSize Property 782

MemData Unit Members 812  
 MemDS Unit Members 846  
 MemUtils Unit Members 870  
 MessageFileLocation Property 616  
 MetaDataKind Property 317  
 MiddleName Property 695  
 Migration Wizard 57  
 MinPoolSize Property 352  
 mmFieldIndex 81  
 mmFieldName 81  
 moCustom 167  
 moDBMonitor 167  
 Mode Property 76  
 moDialog 167  
 Modified Property 732  
 ModifyObject Property 276  
 ModifySQL Property 277  
 ModifyUser Method 671  
 moHandled 167  
 MonitorMessage Method 201  
 moSQLMonitor 167  
 MovedCount Property 77  
 MultiDatabase Property 656

## - N -

Name Property  
     TDAColumn 120  
     TMacro 345  
 NBackupLevel Property 609  
 NBackupOptions Property 609  
 nboNoTriggers 710  
 NetBEUI 593  
 NoOfAttachments Property 638  
 NoReserve Property 766  
 ntBCD 844  
 ntFloat 844  
 ntFmtBCD 844  
 NumberRange Property 291  
 NumBuffers Property 767  
 NumSegments Property 782  
 NumWALBuffers Property 767

## - O -

ObjectType Property 817  
 ODSMajorVersion Property 767

ODSMinorVersion Property 768  
 Offset Property 818  
 Omit Property 151  
 OnAttach Event 605  
 OnBackupProgress Event 113  
 OnBatchMoveProgress Event 79  
 OnConnectionLost Event 203  
 OnError Event  
     TCustomDAConnection 203  
     TDAAlerter 101  
     TDADump 113  
     TDAScript 147  
     TDATransaction 341  
     TIBCTransaction 589  
 OnEvent Event 720  
 OnGetColumnData Event  
     TDALoader 129  
     TIBCLoader 799  
 OnProgress Event 130  
 OnPutData Event  
     TDALoader 130  
     TIBCLoader 799  
 OnRestoreProgress Event 114  
 OnSQL Event 162  
 OnUpdateError Event 868  
 OnUpdateRecord Event 869  
 Options Property  
     TCustomDAConnection 186  
     TCustomDADataset 222  
     TCustomDASQLMonitor 161  
     TCustomIBCDataset 390  
     TDADump 106  
     TIBCBBackupService 614  
     TIBCCConnection 458  
     TIBCRestoreService 662  
     TIBCStatisticalService 684  
     TIBCValidationService 702  
     TVirtualTable 877  
 Overview 1  
 Owner Property 818

## - P -

PageBuffers Property 662  
 PageCache Property 642  
 PageLength Property 642  
 PageSize Property  
     TGDSDatabaseInfo 768

PageSize Property  
     TIBCJournalInformation 642  
     TIBCRestoreService 663  
 ParamByName Method  
     TCustomDADataset 246  
     TCustomDASQL 268  
     TCustomIBCDataSet 403  
     TDAParams 336  
     TIBCCConnection 468  
     TIBCPParams 502  
     TIBCSQL 532  
 ParamCheck Property  
     TCustomDADataset 224  
     TCustomDASQL 261  
 ParamCount Property  
     TCustomDADataset 225  
     TCustomDASQL 261  
 Params Property  
     TCustomDADataset 225  
     TCustomDASQL 261  
     TCustomIBCService 602  
     TDASTatement 151  
     TIBCCConnection 459  
     TIBCScript 805  
     TIBCSQL 526  
     TIBCTransaction 583  
 ParamType Property 329  
 ParamValues Property(Indexer) 262  
 Password Property  
     TCREncryptor 93  
     TCustomDAConnection 187  
     TIBCCConnection 459  
     TIBCUserInfo 696  
 PasswordLabel Property 178  
 Plan Property 391  
 Pooling Property 187  
 PoolingOptions Property 188  
 Port Property 164  
 Prepare Method  
     TCustomDADataset 247  
     TCustomDASQL 269  
     TIBCStoredProc 558  
     TMemDataSet 863  
 Prepared Property  
     TCustomDASQL 263  
     TMemDataSet 852  
 PrepareSQL Method 558  
 ProblemCount Property 77

Protocol Property  
     TCustomIBCService 603  
     TIBCCConnectionOptions 474  
 ProtocolLabel Property 812  
 PurgeCount Property 768  
 PutColumnData Method 127

## - Q -

QueryRecCount Property 291  
 QuoteNames Property  
     TDADatasetOptions 292  
     TDADumpOptions 116

## - R -

Read Method 826  
 ReadArray Method 741  
 ReadArrayItem Method 742  
 ReadArraySlice Method 742  
 ReadBlob Method 787  
 ReadIdxCount Property 769  
 ReadOnly Property  
     TCustomDADataset 226  
     TGDSDatabaseInfo 769  
 Reads Property 769  
 ReadSeqCount Property 769  
 ReclaimMemory Method 627  
 ReconnectTimeout Property 164  
 RecordCount Property 78  
 RecoverTwoPhaseGlobal Property 702  
 RefCount Property 840  
 RefreshObject Property 277  
 RefreshOptions Property 226  
 RefreshRecord Method 247  
 RefreshSQL Property 278  
 Release Method 841  
 ReleaseSavepoint Method 586  
 RemoteDatabasePath Property 656  
 RemoteSite Property 656  
 RemoveConnection Method 586  
 RemoveFromPool Method 201  
 RemoveLicense Method 652  
 RemoveOnRefresh Property 292  
 RemoveTransaction Method 468  
 RequiredFields Property 292  
 Requirements 21

Restore Method	111	SavePassword Property	179
RestoreFromFile Method	111	SaveSQL Method	249
RestoreFromStream Method	112	SaveToFile Method	
RestoreSQL Method	248	TBlob	826
RestoreUpdates Method	864	TVirtualTable	883
Restrictions Property	318	SaveToStream Method	
ResumeTrace Method	690	TBlob	827
Resync Method	248	TVirtualTable	884
Retries Property	178	SaveToXML Method	864
ReturnParams Property	293	Scale Property	818
RevertRecord Method	864	Scan Method	350
rmRaise	359	Script Property	152
rmReconnect	359	SecurityAction Property	666
rmReconnectExecute	359	SecurityDatabaseLocation Property	617
roAfterInsert	358	SendDataSetChangeEventAfterOpen Variable	870
roAfterUpdate	358	Sender Property	794
roBeforeEdit	358	SendEvent Method	
roCreateNewDB	710	TDAAlterer	100
roDeactivateIndexes	710	TIBCAAlterer	719
Role Property	474	SendTimeout Property	165
Rollback Method		Server Property	
TCustomDAConnection	202	TCustomDAConnection	189
TDATransaction	340	TCustomIBCSERVICE	603
TIBCTransaction	587	TIBCCConnection	460
RollbackRetaining Method		ServerImplementation Property	706
TIBCCConnection	469	ServerLabel Property	179
TIBCTransaction	587	ServerVersion Property	706
RollbackSavepoint Method	587	ServiceParamBySPB Property(Indexer)	603
roNoShadow	710	ServiceStart Method	
roNoValidityCheck	710	TCustomIBCSERVICE	605
roOneRelationAtATime	710	TIBCCConfigService	627
roReplace	710	ServiceVersion Property	706
roUseAllSpace	710	SessionName Property	688
roValidationCheck	710	SetAsyncMode Method	627
RowsAffected Property		SetBlobData Method	
TCustomDADataSet	226	TDAParam	333
TCustomDASQL	263	TIBCPParam	499
RowsDeleted Property	392	SetDBSqlDialect Method	628
RowsFetched Property	392	SetFieldsReadOnly Property	293
RowsInserted Property	392	SetFlushInterval Method	628
RowsUpdated Property	393	SetGroupCommit Method	628
		SetItemAsDateTime Method	743
		SetItemAsFloat Method	743
		SetItemAsInteger Method	744
		SetItemAsSmallInt Method	744
		SetItemAsString Method	745
		SetItemAsWideString Method	746
		SetItemsSlice Method	746
 <b>- S -</b>			
saAddUser	711		
saDeleteUser	711		
saDisplayUser	711		
saModifyUser	711		

- SetItemValue Method 747
- SetKey Method 93
- SetLingerInterval Method 629
- SetOrderBy Method 249
- SetPageBuffers Method 629
- SetReadOnly Method 630
- SetReclaimInterval Method 630
- SetReserveSpace Method 631
- SetSweepInterval Method 631
- ShutdownDatabase Method 632
- Size Property
  - TAttribute 819
  - TBlob 823
  - TDAParam 329
  - TObjectType 836
- soDataPages 712
- soDbLog 712
- soHeaderPages 712
- soIndexPages 712
- soRecordVersions 712
- soStatTables 712
- soSystemRelations 712
- Source Property 78
- SPX 593
- SQL Property
  - TCustomDADataset 227
  - TCustomDASQL 264
  - TDADump 107
  - TDAScript 140
  - TDASTatement 152
  - TIBCConnection 460
- SQL Property(Indexer) 278
- SQLDelete Property 227
- SQLDialect Property 461
- SQLErrorMsg Property 794
- SQLInsert Property 228
- SQLLock Property 229
- SQLRefresh Property 229
- SQLRole Property 696
- SQLSaved Method 250
- SQLType Property
  - TCustomIBCDataSet 393
  - TIBCSQL 527
- SQLUpdate Property 230
- Start Method 100
- StartLine Property
  - TDAScript 140
  - TDASTatement 152
- StartOffset Property
  - TDAScript 140
  - TDASTatement 153
- StartPos Property
  - TDAScript 141
  - TDASTatement 153
- StartSavepoint Method 588
- StartTrace Method 690
- StartTransaction Method
  - TCustomDAConnection 202
  - TDATransaction 340
- State Property 656
- Statements Property 141
- stBinary 845
- stCaseInsensitive 845
- stCaseSensitive 845
- Stop Method 101
- StopTrace Method 691
- StoredProcName Property 549
- StoreLogInfo Property 179
- Streamed Property 783
- StreamedBlobs Property 485
- StrictUpdate Property
  - TDADatasetOptions 293
  - TIBCDataSetOptions 485
- SubType Property 783
- SuspendEUA Method 671
- SuspendTrace Method 691
- SweepDatabase Method
  - TIBCConfigService 632
  - TIBCValidationService 704
- SweepInterval Property 770
- SystemUserName Property 696

## - T -

- TableName Property
  - TCustomIBCArray 733
  - TDALoader 124
  - TIBCLoader 798
  - TIBCTable 576
- TableNames Property
  - TDADump 107
  - TIBCStatisticalService 684
- taCommit 69
- taCommitAdvise 712
- taCommitRetaining 593
- TAfterExecuteEvent Procedure Reference 354

- TAfterFetchEvent Procedure Reference 354
- TAfterStatementExecuteEvent Procedure Reference 156
- TAlterErrorEvent Procedure Reference 102
- taRollback 69
- taRollbackAdvise 712
- taRollbackRetaining 593
- TAttribute Class 814
- TAttribute.AttributeNo Property 816
- TAttribute.DataSize Property 816
- TAttribute.DataType Property 817
- TAttribute.Length Property 817
- TAttribute.ObjectType Property 817
- TAttribute.Offset Property 818
- TAttribute.Owner Property 818
- TAttribute.Scale Property 818
- TAttribute.Size Property 819
- taUnknownAdvise 712
- TBeforeFetchEvent Procedure Reference 354
- TBeforeFetchProc Procedure Reference 67
- TBeforeStatementExecuteEvent Procedure Reference 156
- TBlob Class 819
- TBlob.Assign Method 824
- TBlob.AsString Property 821
- TBlob.AsWideString Property 822
- TBlob.Clear Method 824
- TBlob.IsUnicode Property 822
- TBlob.LoadFromFile Method 825
- TBlob.LoadFromStream Method 825
- TBlob.Read Method 826
- TBlob.SaveToFile Method 826
- TBlob.SaveToStream Method 827
- TBlob.Size Property 823
- TBlob.Truncate Method 827
- TBlob.Write Method 828
- TCompressBlobMode Enumeration 842
- TCompressedBlob Class 828
- TCompressedBlob.Compressed Property 832
- TCompressedBlob.CompressedSize Property 832
- TConnectionLostEvent Procedure Reference 355
- TConnLostCause Enumeration 843
- TCP 593
- TCRBatchMode Enumeration 81
- TCRBatchMove Class 70
- TCRBatchMove.AbortOnKeyViol Property 73
- TCRBatchMove.AbortOnProblem Property 74
- TCRBatchMove.ChangedCount Property 74
- TCRBatchMove.CommitCount Property 74
- TCRBatchMove.Destination Property 75
- TCRBatchMove.Execute Method 79
- TCRBatchMove.FieldMappingMode Property 75
- TCRBatchMove.KeyViolCount Property 75
- TCRBatchMove.Mappings Property 76
- TCRBatchMove.Mode Property 76
- TCRBatchMove.MovedCount Property 77
- TCRBatchMove.OnBatchMoveProgress Event 79
- TCRBatchMove.ProblemCount Property 77
- TCRBatchMove.RecordCount Property 78
- TCRBatchMove.Source Property 78
- TCRBatchMoveProgressEvent Procedure Reference 80
- TCRCursor Class 66
- TCRDataSource Class 174
- TCREncDataHeader Enumeration 94
- TCREncryptionAlgorithm Enumeration 95
- TCREncryptor Class 90
- TCREncryptor.DataHeader Property 91
- TCREncryptor.EncryptionAlgorithm Property 91
- TCREncryptor.HashAlgorithm Property 92
- TCREncryptor.InvalidHashAction Property 92
- TCREncryptor.Password Property 93
- TCREncryptor.SetKey Method 93
- TCRFieldMappingMode Enumeration 81
- TCRHashAlgorithm Enumeration 95
- TCRInvalidHashAction Enumeration 96
- TCRIsoLevel Enumeration 68
- TCRTransactionAction Enumeration 69
- TCustomConnectDialog Class 174
- TCustomConnectDialog.CancelButton Property 176
- TCustomConnectDialog.Caption Property 177
- TCustomConnectDialog.ConnectButton Property 177
- TCustomConnectDialog.DialogClass Property 177
- TCustomConnectDialog.Execute Method 180
- TCustomConnectDialog.GetServerList Method 181
- TCustomConnectDialog.LabelSet Property 178
- TCustomConnectDialog.PasswordLabel Property 178
- TCustomConnectDialog.Retries Property 178
- TCustomConnectDialog.SavePassword Property 179
- TCustomConnectDialog.ServerLabel Property 179
- TCustomConnectDialog.StoreLogInfo Property 179
- TCustomConnectDialog.UsernameLabel Property 180



- TCustomDAConnection Class 181
- TCustomDAConnection.ApplyUpdates Method 191
- TCustomDAConnection.Commit Method 192
- TCustomDAConnection.Connect Method 192
- TCustomDAConnection.ConnectDialog Property 184
- TCustomDAConnection.ConvertEOL Property 185
- TCustomDAConnection.CreateDataSet Method 193
- TCustomDAConnection.CreateSQL Method 193
- TCustomDAConnection.Disconnect Method 194
- TCustomDAConnection.ExecProc Method 194
- TCustomDAConnection.ExecProcEx Method 196
- TCustomDAConnection.ExecSQL Method 197
- TCustomDAConnection.ExecSQLEx Method 198
- TCustomDAConnection.GetDatabaseNames Method 199
- TCustomDAConnection.GetStoredProcNames Method 199
- TCustomDAConnection.GetTableNames Method 200
- TCustomDAConnection.InTransaction Property 185
- TCustomDAConnection.LoginPrompt Property 185
- TCustomDAConnection.MonitorMessage Method 201
- TCustomDAConnection.OnConnectionLost Event 203
- TCustomDAConnection.OnError Event 203
- TCustomDAConnection.Options Property 186
- TCustomDAConnection.Password Property 187
- TCustomDAConnection.Pooling Property 187
- TCustomDAConnection.PoolingOptions Property 188
- TCustomDAConnection.RemoveFromPool Method 201
- TCustomDAConnection.Rollback Method 202
- TCustomDAConnection.Server Property 189
- TCustomDAConnection.StartTransaction Method 202
- TCustomDAConnection.Username Property 189
- TCustomDADataset Class 204
- TCustomDADataset.AddWhere Method 235
- TCustomDADataset.AfterExecute Event 253
- TCustomDADataset.AfterFetch Event 253
- TCustomDADataset.AfterUpdateExecute Event 253
- TCustomDADataset.BaseSQL Property 215
- TCustomDADataset.BeforeFetch Event 254
- TCustomDADataset.BeforeUpdateExecute Event 254
- TCustomDADataset.BreakExec Method 236
- TCustomDADataset.Connection Property 215
- TCustomDADataset.CreateBlobStream Method 236
- TCustomDADataset.Debug Property 216
- TCustomDADataset.DeleteWhere Method 236
- TCustomDADataset.DetailFields Property 216
- TCustomDADataset.Disconnected Property 217
- TCustomDADataset.Encryption Property 217
- TCustomDADataset.Execute Method 237
- TCustomDADataset.Executing Method 237
- TCustomDADataset.Fetched Method 238
- TCustomDADataset.Fetching Method 238
- TCustomDADataset.FetchingAll Method 239
- TCustomDADataset.FetchRows Property 217
- TCustomDADataset.FilterSQL Property 218
- TCustomDADataset.FinalSQL Property 218
- TCustomDADataset.FindKey Method 239
- TCustomDADataset.FindMacro Method 240
- TCustomDADataset.FindNearest Method 240
- TCustomDADataset.FindParam Method 241
- TCustomDADataset.GetDataType Method 241
- TCustomDADataset.GetFieldObject Method 242
- TCustomDADataset.GetFieldPrecision Method 242
- TCustomDADataset.GetFieldScale Method 243
- TCustomDADataset.GetOrderBy Method 244
- TCustomDADataset.GotoCurrent Method 244
- TCustomDADataset.IsQuery Property 219
- TCustomDADataset.KeyFields Property 219
- TCustomDADataset.Lock Method 245
- TCustomDADataset.MacroByName Method 245
- TCustomDADataset.MacroCount Property 220
- TCustomDADataset.Macros Property 220
- TCustomDADataset.MasterFields Property 221
- TCustomDADataset.MasterSource Property 221
- TCustomDADataset.Options Property 222
- TCustomDADataset.ParamByName Method 246
- TCustomDADataset.ParamCheck Property 224
- TCustomDADataset.ParamCount Property 225
- TCustomDADataset.Params Property 225
- TCustomDADataset.Prepare Method 247
- TCustomDADataset.ReadOnly Property 226
- TCustomDADataset.RefreshOptions Property 226
- TCustomDADataset.RefreshRecord Method 247
- TCustomDADataset.RestoreSQL Method 248
- TCustomDADataset.Resync Method 248

TCustomDADataSet.RowsAffected Property	226	TCustomDAUpdateSQL.DeleteSQL Property	274
TCustomDADataSet.SaveSQL Method	249	TCustomDAUpdateSQL.ExecSQL Method	279
TCustomDADataSet.SetOrderBy Method	249	TCustomDAUpdateSQL.InsertObject Property	275
TCustomDADataSet.SQL Property	227	TCustomDAUpdateSQL.InsertSQL Property	275
TCustomDADataSet.SQLDelete Property	227	TCustomDAUpdateSQL.LockObject Property	276
TCustomDADataSet.SQLInsert Property	228	TCustomDAUpdateSQL.LockSQL Property	276
TCustomDADataSet.SQLLock Property	229	TCustomDAUpdateSQL.ModifyObject Property	276
TCustomDADataSet.SQLRefresh Property	229	TCustomDAUpdateSQL.ModifySQL Property	277
TCustomDADataSet.SQLSaved Method	250	TCustomDAUpdateSQL.RefreshObject Property	277
TCustomDADataSet.SQLUpdate Property	230	TCustomDAUpdateSQL.RefreshSQL Property	278
TCustomDADataSet.UniDirectional Property	231	TCustomDAUpdateSQL.SQL Property(Indexer)	278
TCustomDADataSet.Unlock Method	250	TCustomIBCArray Class	721
TCustomDASQL Class	255	TCustomIBCArray.ArrayDimensions Property	726
TCustomDASQL.AfterExecute Event	271	TCustomIBCArray.ArrayHighBound Property(Indexer)	726
TCustomDASQL.ChangeCursor Property	258	TCustomIBCArray.ArrayID Property	727
TCustomDASQL.Connection Property	258	TCustomIBCArray.ArrayLowBound Property(Indexer)	727
TCustomDASQL.Debug Property	259	TCustomIBCArray.ArraySize Property	728
TCustomDASQL.Execute Method	265	TCustomIBCArray.Assign Method	735
TCustomDASQL.Executing Method	266	TCustomIBCArray.AsString Property	728
TCustomDASQL.FinalSQL Property	259	TCustomIBCArray.Cached Property	728
TCustomDASQL.FindMacro Method	266	TCustomIBCArray.CachedDimensions Property	729
TCustomDASQL.FindParam Method	267	TCustomIBCArray.CachedHighBound Property(Indexer)	729
TCustomDASQL.MacroByName Method	267	TCustomIBCArray.CachedLowBound Property(Indexer)	729
TCustomDASQL.MacroCount Property	260	TCustomIBCArray.CachedSize Property	730
TCustomDASQL.Macros Property	260	TCustomIBCArray.ClearArray Method	735
TCustomDASQL.ParamByName Method	268	TCustomIBCArray.ColumnName Property	730
TCustomDASQL.ParamCheck Property	261	TCustomIBCArray.CreateTemporaryArray Method	736
TCustomDASQL.ParamCount Property	261	TCustomIBCArray.DbHandle Property	731
TCustomDASQL.Params Property	261	TCustomIBCArray.GetArrayInfo Method	736
TCustomDASQL.ParamValues Property(Indexer)	262	TCustomIBCArray.GetItemAsDateTime Method	737
TCustomDASQL.Prepare Method	269	TCustomIBCArray.GetItemAsFloat Method	737
TCustomDASQL.Prepared Property	263	TCustomIBCArray.GetItemAsInteger Method	738
TCustomDASQL.RowsAffected Property	263	TCustomIBCArray.GetItemAsSmallInt Method	738
TCustomDASQL.SQL Property	264	TCustomIBCArray.GetItemAsString Method	739
TCustomDASQL.UnPrepare Method	269	TCustomIBCArray.GetItemAsWideString Method	740
TCustomDASQL.WaitExecuting Method	270	TCustomIBCArray.GetItemsSlice Method	740
TCustomDASQLMonitor Class	159	TCustomIBCArray.GetItemValue Method	741
TCustomDASQLMonitor.Active Property	160	TCustomIBCArray.IsNull Property	731
TCustomDASQLMonitor.DBMonitorOptions Property	160	TCustomIBCArray.Items Property	731
TCustomDASQLMonitor.OnSQL Event	162		
TCustomDASQLMonitor.Options Property	161		
TCustomDASQLMonitor.TraceFlags Property	161		
TCustomDAUpdateSQL Class	271		
TCustomDAUpdateSQL.Apply Method	279		
TCustomDAUpdateSQL.DataSet Property	274		
TCustomDAUpdateSQL.DeleteObject Property	274		



- 
- TCustomIBCArray.ItemScale Property 732
  - TCustomIBCArray.ItemSize Property 732
  - TCustomIBCArray.Modified Property 732
  - TCustomIBCArray.ReadArray Method 741
  - TCustomIBCArray.ReadArrayItem Method 742
  - TCustomIBCArray.ReadArraySlice Method 742
  - TCustomIBCArray.SetItemAsDateTime Method 743
  - TCustomIBCArray.SetItemAsFloat Method 743
  - TCustomIBCArray.SetItemAsInteger Method 744
  - TCustomIBCArray.SetItemAsSmallInt Method 744
  - TCustomIBCArray.SetItemAsString Method 745
  - TCustomIBCArray.SetItemAsWideString Method 746
  - TCustomIBCArray.SetItemsSlice Method 746
  - TCustomIBCArray.SetItemValue Method 747
  - TCustomIBCArray.TableName Property 733
  - TCustomIBCArray.TrHandle Property 733
  - TCustomIBCArray.WriteArray Method 747
  - TCustomIBCArray.WriteArraySlice Method 748
  - TCustomIBCDataset Class 370
  - TCustomIBCDataset.AutoCommit Property 386
  - TCustomIBCDataset.Connection Property 386
  - TCustomIBCDataset.CreateProcCall Method 401
  - TCustomIBCDataset.Cursor Property 386
  - TCustomIBCDataset.DMLRefresh Property 387
  - TCustomIBCDataset.Fetched Method 401
  - TCustomIBCDataset.FindParam Method 401
  - TCustomIBCDataset.GeneratorMode Property 388
  - TCustomIBCDataset.GeneratorStep Property 388
  - TCustomIBCDataset.GetArray Method 402
  - TCustomIBCDataset.GetBlob Method 403
  - TCustomIBCDataset.Handle Property 388
  - TCustomIBCDataset.IsQuery Property 389
  - TCustomIBCDataset.KeyGenerator Property 389
  - TCustomIBCDataset.LockMode Property 390
  - TCustomIBCDataset.Options Property 390
  - TCustomIBCDataset.ParamByName Method 403
  - TCustomIBCDataset.Plan Property 391
  - TCustomIBCDataset.RowsDeleted Property 392
  - TCustomIBCDataset.RowsFetched Property 392
  - TCustomIBCDataset.RowsInserted Property 392
  - TCustomIBCDataset.RowsUpdated Property 393
  - TCustomIBCDataset.SQLType Property 393
  - TCustomIBCDataset.Transaction Property 393
  - TCustomIBCDataset.UpdateTransaction Property 394
  - TCustomIBCQuery Class 404
  - TCustomIBCSERVICE Class 600
  - TCustomIBCSERVICE.Active Property 602
  - TCustomIBCSERVICE.Attach Method 604
  - TCustomIBCSERVICE.Detach Method 604
  - TCustomIBCSERVICE.Handle Property 602
  - TCustomIBCSERVICE.LoginPrompt Property 602
  - TCustomIBCSERVICE.OnAttach Event 605
  - TCustomIBCSERVICE.Params Property 602
  - TCustomIBCSERVICE.Protocol Property 603
  - TCustomIBCSERVICE.Server Property 603
  - TCustomIBCSERVICE.ServiceParamBySPB Property(Indexer) 603
  - TCustomIBCSERVICE.ServiceStart Method 605
  - TCustomIBCTable Class 413
  - TCustomIBCTable.DeleteTable Method 437
  - TCustomIBCTable.EmptyTable Method 437
  - TCustomIBCTable.Exists Property 429
  - TDAAlerter Class 97
  - TDAAlerter.Active Property 98
  - TDAAlerter.AutoRegister Property 99
  - TDAAlerter.Connection Property 99
  - TDAAlerter.OnError Event 101
  - TDAAlerter.SendEvent Method 100
  - TDAAlerter.Start Method 100
  - TDAAlerter.Stop Method 101
  - TDABackupProgressEvent Procedure Reference 117
  - TDAColumn Class 118
  - TDAColumn.FieldType Property 119
  - TDAColumn.Name Property 120
  - TDAColumns Class 120
  - TDAColumns.Items Property(Indexer) 121
  - TDAConnectionErrorEvent Procedure Reference 355
  - TDAConnectionOptions Class 280
  - TDAConnectionOptions.DefaultSortType Property 282
  - TDAConnectionOptions.DisconnectedMode Property 282
  - TDAConnectionOptions.KeepDesignConnected Property 282
  - TDAConnectionOptions.LocalFailover Property 283
  - TDADatasetOptions Class 283
  - TDADatasetOptions.AutoPrepare Property 288
  - TDADatasetOptions.CacheCalcFields Property 288
  - TDADatasetOptions.CompressBlobMode Property 288
  - TDADatasetOptions.DefaultValues Property 289
  - TDADatasetOptions.DetailDelay Property 289
  - TDADatasetOptions.FieldsOrigin Property 289

- TDADatasetOptions.FlatBuffers Property 290
- TDADatasetOptions.LocalMasterDetail Property 290
- TDADatasetOptions.LongStrings Property 290
- TDADatasetOptions.NumberRange Property 291
- TDADatasetOptions.QueryRecCount Property 291
- TDADatasetOptions.QuoteNames Property 292
- TDADatasetOptions.RemoveOnRefresh Property 292
- TDADatasetOptions.RequiredFields Property 292
- TDADatasetOptions.ReturnParams Property 293
- TDADatasetOptions.SetFieldsReadOnly Property 293
- TDADatasetOptions.StrictUpdate Property 293
- TDADatasetOptions.TrimFixedChar Property 294
- TDADatasetOptions.UpdateAllFields Property 294
- TDADatasetOptions.UpdateBatchSize Property 294
- TDADump Class 103
- TDADump.Backup Method 108
- TDADump.BackupQuery Method 109
- TDADump.BackupToFile Method 109
- TDADump.BackupToStream Method 110
- TDADump.Connection Property 105
- TDADump.Debug Property 106
- TDADump.OnBackupProgress Event 113
- TDADump.OnError Event 113
- TDADump.OnRestoreProgress Event 114
- TDADump.Options Property 106
- TDADump.Restore Method 111
- TDADump.RestoreFromFile Method 111
- TDADump.RestoreFromStream Method 112
- TDADump.SQL Property 107
- TDADump.TableNames Property 107
- TDADumpOptions Class 114
- TDADumpOptions.AddDrop Property 115
- TDADumpOptions.GenerateHeader Property 116
- TDADumpOptions.QuoteNames Property 116
- TDAEncryptionOptions Class 295
- TDAEncryptionOptions.Encryptor Property 296
- TDAEncryptionOptions.Fields Property 296
- TDALoader Class 122
- TDALoader.Columns Property 124
- TDALoader.Connection Property 124
- TDALoader.CreateColumns Method 125
- TDALoader.Load Method 126
- TDALoader.LoadFromDataSet Method 126
- TDALoader.OnGetColumnData Event 129
- TDALoader.OnProgress Event 130
- TDALoader.OnPutData Event 130
- TDALoader.PutColumnData Method 127
- TDALoader.TableName Property 124
- TDAMapRule Class 297
- TDAMapRule.DBLengthMax Property 299
- TDAMapRule.DBLengthMin Property 299
- TDAMapRule.DBScaleMax Property 299
- TDAMapRule.DBScaleMin Property 300
- TDAMapRule.DBType Property 300
- TDAMapRule.FieldLength Property 300
- TDAMapRule.FieldName Property 301
- TDAMapRule.FieldScale Property 301
- TDAMapRule.FieldType Property 301
- TDAMapRule.IgnoreErrors Property 302
- TDAMapRules Class 302
- TDAMapRules.AddDBTypeRule Method 304
- TDAMapRules.AddFieldRule Method 309
- TDAMapRules.AddRule Method 311
- TDAMetaData Class 312
- TDAMetaData.Connection Property 317
- TDAMetaData.GetMetaDataKinds Method 321
- TDAMetaData.GetRestrictions Method 321
- TDAMetaData.MetaDataKind Property 317
- TDAMetaData.Restrictions Property 318
- TDANumericType Enumeration 844
- TDAParam Class 322
- TDAParam.AsBlob Property 325
- TDAParam.AsBlobRef Property 325
- TDAParam.AsFloat Property 326
- TDAParam.AsInteger Property 326
- TDAParam.AsLargeInt Property 326
- TDAParam.AsMemo Property 327
- TDAParam.AsMemoRef Property 327
- TDAParam.AssignField Method 331
- TDAParam.AssignFieldValue Method 331
- TDAParam.AsSQLTimeStamp Property 327
- TDAParam.AsString Property 328
- TDAParam.AsWideString Property 328
- TDAParam.DataType Property 328
- TDAParam.IsNull Property 329
- TDAParam.LoadFromFile Method 332
- TDAParam.LoadFromStream Method 332
- TDAParam.ParamType Property 329
- TDAParam.SetBlobData Method 333
- TDAParam.Size Property 329
- TDAParam.Value Property 330
- TDAParams Class 334
- TDAParams.FindParam Method 336

- TDAParams.Items Property(Indexer) 335
- TDAParams.ParamByName Method 336
- TDAPutDataEvent Procedure Reference 131
- TDARestoreProgressEvent Procedure Reference 117
- TDAScript Class 133
- TDAScript.AfterExecute Event 147
- TDAScript.BeforeExecute Event 147
- TDAScript.BreakExec Method 142
- TDAScript.Connection Property 137
- TDAScript.DataSet Property 137
- TDAScript.Debug Property 138
- TDAScript.Delimiter Property 138
- TDAScript.EndLine Property 138
- TDAScript.EndOffset Property 139
- TDAScript.EndPos Property 139
- TDAScript.ErrorOffset Method 143
- TDAScript.Execute Method 143
- TDAScript.ExecuteFile Method 144
- TDAScript.ExecuteNext Method 144
- TDAScript.ExecuteStream Method 145
- TDAScript.FindMacro Method 145
- TDAScript.MacroByName Method 146
- TDAScript.Macros Property 139
- TDAScript.OnError Event 147
- TDAScript.SQL Property 140
- TDAScript.StartLine Property 140
- TDAScript.StartOffset Property 140
- TDAScript.StartPos Property 141
- TDAScript.Statements Property 141
- TDASTatement Class 148
- TDASTatement.EndLine Property 150
- TDASTatement.EndOffset Property 150
- TDASTatement.EndPos Property 151
- TDASTatement.Execute Method 153
- TDASTatement.Omit Property 151
- TDASTatement.Params Property 151
- TDASTatement.Script Property 152
- TDASTatement.SQL Property 152
- TDASTatement.StartLine Property 152
- TDASTatement.StartOffset Property 153
- TDASTatement.StartPos Property 153
- TDASTatements Class 154
- TDASTatements.Items Property(Indexer) 155
- TDATraceFlag Enumeration 167
- TDATraceFlags Set 165
- TDATransaction Class 337
- TDATransaction.Active Property 338
- TDATransaction.Commit Method 339
- TDATransaction.DefaultCloseAction Property 339
- TDATransaction.OnError Event 341
- TDATransaction.Rollback Method 340
- TDATransaction.StartTransaction Method 340
- TDATransactionErrorEvent Procedure Reference 356
- TDBMonitorOptions Class 162
- TDBMonitorOptions.Host Property 164
- TDBMonitorOptions.Port Property 164
- TDBMonitorOptions.ReconnectTimeout Property 164
- TDBMonitorOptions.SendTimeout Property 165
- TDBObject Class 832
- TErrorAction Enumeration 157
- tfBlob 167
- tfConnect 167
- tfError 167
- tfMisc 167
- tfObjDestroy 167
- tfParams 167
- tfPool 167
- tfQExecute 167
- tfQFetch 167
- tfQPrepare 167
- tfService 167
- tfStmt 167
- tfTransact 167
- tgCommitGlobal 713
- TGDSDatabaseInfo Class 754
- TGDSDatabaseInfo.Allocation Property 760
- TGDSDatabaseInfo.AttachmentID Property 760
- TGDSDatabaseInfo.BackoutCount Property 761
- TGDSDatabaseInfo.BaseLevel Property 761
- TGDSDatabaseInfo.CurLogFileName Property 761
- TGDSDatabaseInfo.CurLogPartitionOffset Property 762
- TGDSDatabaseInfo.CurrentMemory Property 762
- TGDSDatabaseInfo.DBFileName Property 762
- TGDSDatabaseInfo.DBImplementationClass Property 763
- TGDSDatabaseInfo.DBImplementationNo Property 763
- TGDSDatabaseInfo.DBSiteName Property 763
- TGDSDatabaseInfo.DeleteCount Property 763
- TGDSDatabaseInfo.ExpungeCount Property 764
- TGDSDatabaseInfo.Fetches Property 764
- TGDSDatabaseInfo.ForcedWrites Property 764
- TGDSDatabaseInfo.InsertCount Property 765

TGDSDatabaseInfo.IsRemoteConnect Property	765	TIBCAlerter.SendEvent Method	719
TGDSDatabaseInfo.LogFile Property	765	TIBCAlerter.Transaction Property	718
TGDSDatabaseInfo.Marks Property	766	TIBCAAlertEvent Procedure Reference	720
TGDSDatabaseInfo.MaxMemory Property	766	TIBCArrary Class	437
TGDSDatabaseInfo.NoReserve Property	766	TIBCArrary.ItemType Property	443
TGDSDatabaseInfo.NumBuffers Property	767	TIBCArraryField Class	444
TGDSDatabaseInfo.NumWALBuffers Property	767	TIBCArraryField.AsArray Property	445
TGDSDatabaseInfo.ODSMajorVersion Property	767	TIBCBBackupOption Enumeration	709
TGDSDatabaseInfo.ODSMinorVersion Property	768	TIBCBBackupOptions Set	707
TGDSDatabaseInfo.PageSize Property	768	TIBCBBackupRestoreService Class	606
TGDSDatabaseInfo.PurgeCount Property	768	TIBCBBackupRestoreService.BackupFile Property	608
TGDSDatabaseInfo.ReadIdxCount Property	769	TIBCBBackupRestoreService.NBackupLevel Property	609
TGDSDatabaseInfo.ReadOnly Property	769	TIBCBBackupRestoreService.NBackupOptions Property	609
TGDSDatabaseInfo.Reads Property	769	TIBCBBackupRestoreService.UseNBackup Property	609
TGDSDatabaseInfo.ReadSeqCount Property	769	TIBCBBackupRestoreService.Verbose Property	610
TGDSDatabaseInfo.SweepInterval Property	770	TIBCBBackupService Class	610
TGDSDatabaseInfo.UpdateCount Property	770	TIBCBBackupService.BackupFile Property	613
TGDSDatabaseInfo.UserNames Property	770	TIBCBBackupService.BlockingFactor Property	613
TGDSDatabaseInfo.Version Property	771	TIBCBBackupService.Database Property	614
TGDSDatabaseInfo.WALAveragGroupCommitSize Property	771	TIBCBBackupService.Options Property	614
TGDSDatabaseInfo.WALAveragelIOSize Property	771	TIBCBBackupService.Verbose Property	614
TGDSDatabaseInfo.WALBufferSize Property	772	TIBCBlob Class	775
TGDSDatabaseInfo.WALCheckpointLength Property	772	TIBCBlob.AllocBlob Method	785
TGDSDatabaseInfo.WALCurCheckpointInterval Property	772	TIBCBlob.Cached Property	779
TGDSDatabaseInfo.WALGroupCommitWaitUSecs Property	773	TIBCBlob.CharsetID Property	779
TGDSDatabaseInfo.WALNumCommits Property	773	TIBCBlob.CloseBlob Method	786
TGDSDatabaseInfo.WALNumIO Property	773	TIBCBlob.ConversionCharsetID Property	780
TGDSDatabaseInfo.WALPrvCheckpointFilename Property	774	TIBCBlob.ConversionSubType Property	780
TGDSDatabaseInfo.WALPrvCheckpointPartOffset Property	774	TIBCBlob.DbHandle Property	781
TGDSDatabaseInfo.Writes Property	774	TIBCBlob.FreeBlob Method	786
TGeneratorMode Enumeration	592	TIBCBlob.Handle Property	781
TGetColumnDataEvent Procedure Reference	131	TIBCBlob.ID Property	782
tgNoGlobalAction	713	TIBCBlob.IsInit Method	786
tgRollbackGlobal	713	TIBCBlob.LengthBlob Method	787
TIBCAlerter Class	715	TIBCBlob.MaxSegmentSize Property	782
TIBCAlerter.AutoCommit Property	717	TIBCBlob.NumSegments Property	782
TIBCAlerter.Connection Property	717	TIBCBlob.ReadBlob Method	787
TIBCAlerter.Events Property	717	TIBCBlob.Streamed Property	783
TIBCAlerter.OnEvent Event	720	TIBCBlob.SubType Property	783
		TIBCBlob.TrHandle Property	784
		TIBCBlob.WriteBlob Method	787
		TIBCCConfigParams Class	615
		TIBCCConfigParams.BaseLocation Property	616
		TIBCCConfigParams.LockFileLocation Property	616



- TIBCCfgParams.MessageFileLocation Property 616  
 TIBCCfgParams.SecurityDatabaseLocation Property 617  
 TIBCCfgService Class 617  
 TIBCCfgService.ActivateShadow Method 623  
 TIBCCfgService.AlterJournal Method 624  
 TIBCCfgService.BringDatabaseOnline Method 624  
 TIBCCfgService.Connection Property 620  
 TIBCCfgService.CreateJournal Method 624  
 TIBCCfgService.CreateJournalArchive Method 625  
 TIBCCfgService.Database Property 621  
 TIBCCfgService.DisableFlush Method 625  
 TIBCCfgService.DropJournal Method 625  
 TIBCCfgService.DropJournalArchive Method 626  
 TIBCCfgService.FlushDatabase Method 626  
 TIBCCfgService.GetJournalInformation Method 626  
 TIBCCfgService.JournalInformation Property 621  
 TIBCCfgService.ReclaimMemory Method 627  
 TIBCCfgService.ServiceStart Method 627  
 TIBCCfgService.SetAsyncMode Method 627  
 TIBCCfgService.SetDBSqlDialect Method 628  
 TIBCCfgService.SetFlushInterval Method 628  
 TIBCCfgService.SetGroupCommit Method 628  
 TIBCCfgService.SetLingerInterval Method 629  
 TIBCCfgService.SetPageBuffers Method 629  
 TIBCCfgService.SetReadOnly Method 630  
 TIBCCfgService.SetReclaimInterval Method 630  
 TIBCCfgService.SetReserveSpace Method 631  
 TIBCCfgService.SetSweepInterval Method 631  
 TIBCCfgService.ShutdownDatabase Method 632  
 TIBCCfgService.SweepDatabase Method 632  
 TIBCCfgService.Transaction Property 621  
 TIBCCfgService.Connection Class 808  
 TIBCCfgService.Connection Property 811  
 TIBCCfgService.DatabaseLabel Property 812  
 TIBCCfgService.ProtocolLabel Property 812  
 TIBCCfgService Class 445  
 TIBCCfgService.AddTransaction Method 465  
 TIBCCfgService.AssignConnect Method 466  
 TIBCCfgService.AutoCommit Property 452  
 TIBCCfgService.ClientLibrary Property 454  
 TIBCCfgService.CommitRetaining Method 466  
 TIBCCfgService.Connected Property 454  
 TIBCCfgService.ConnectPrompt Property 454  
 TIBCCfgService.CreateDatabase Method 467  
 TIBCCfgService.Database Property 455  
 TIBCCfgService.DatabaseInfo Property 456  
 TIBCCfgService.DBSQLDialect Property 456  
 TIBCCfgService.Debug Property 456  
 TIBCCfgService.DefaultTransaction Property 457  
 TIBCCfgService.DropDatabase Method 467  
 TIBCCfgService.FindDefaultTransaction Method 467  
 TIBCCfgService.Handle Property 457  
 TIBCCfgService.LastError Property 458  
 TIBCCfgService.Options Property 458  
 TIBCCfgService.ParamByName Method 468  
 TIBCCfgService.Params Property 459  
 TIBCCfgService.Password Property 459  
 TIBCCfgService.RemoveTransaction Method 468  
 TIBCCfgService.RollbackRetaining Method 469  
 TIBCCfgService.Server Property 460  
 TIBCCfgService.SQL Property 460  
 TIBCCfgService.SQLDialect Property 461  
 TIBCCfgService.TransactionCount Property 461  
 TIBCCfgService.Transactions Property(Indexer) 462  
 TIBCCfgService.Username Property 462  
 TIBCCfgServiceOptions Class 469  
 TIBCCfgServiceOptions.CharLength Property 472  
 TIBCCfgServiceOptions.Charset Property 472  
 TIBCCfgServiceOptions.EnableBCD Property 473  
 TIBCCfgServiceOptions.EnableFMTBCD Property 473  
 TIBCCfgServiceOptions.EnableMemos Property 473  
 TIBCCfgServiceOptions.Protocol Property 474  
 TIBCCfgServiceOptions.Role Property 474  
 TIBCCfgServiceOptions.UseUnicode Property 474  
 TIBCCfgServicePoolManager Class 791  
 TIBCCfgServiceAndQueryService Class 632  
 TIBCCfgServiceAndQueryService.BufferSize Property 635  
 TIBCCfgServiceAndQueryService.Eof Property 635  
 TIBCCfgServiceAndQueryService.GetNextChunk Method 636  
 TIBCCfgServiceAndQueryService.GetNextLine Method 637  
 TIBCCfgServiceInfo Class 637  
 TIBCCfgServiceInfo.DbName Property 638  
 TIBCCfgServiceInfo.NoOfAttachments Property 638

- © 2013 *Enter your company name*

- TIBCSript.AutoDDL Property 804  
TIBCSript.Connection Property 805  
TIBCSript.DataSet Property 805  
TIBCSript.Params Property 805  
TIBCSript.Transaction Property 806  
TIBCSecurityAction Enumeration 711  
TIBCSecurityService Class 663  
TIBCSecurityService.AddUser Method 669  
TIBCSecurityService.DeleteUser Method 670  
TIBCSecurityService.DisplayUser Method 670  
TIBCSecurityService.DisplayUsers Method 670  
TIBCSecurityService.EnableEUA Method 671  
TIBCSecurityService.ModifyUser Method 671  
TIBCSecurityService.SecurityAction Property 666  
TIBCSecurityService.SuspendEUA Method 671  
TIBCSecurityService.UserDatabase Property 667  
TIBCSecurityService.UserInfo Property 667  
TIBCSecurityService.UserInfos Property(Indexer) 667  
TIBCSecurityService.UserInfosCount Property 668  
TIBCServerProperties Class 672  
TIBCServerProperties.AddAlias Method 678  
TIBCServerProperties.AliasCount Property 675  
TIBCServerProperties.ConfigParams Property 675  
TIBCServerProperties.DatabaseInfo Property 675  
TIBCServerProperties.DeleteAlias Method 678  
TIBCServerProperties.Fetch Method 679  
TIBCServerProperties.FetchAliasInfo Method 679  
TIBCServerProperties.FetchConfigParams Method 679  
TIBCServerProperties.FetchDatabaseInfo Method 680  
TIBCServerProperties.FetchLicenseInfo Method 680  
TIBCServerProperties.FetchLicenseMaskInfo Method 680  
TIBCServerProperties.FetchVersionInfo Method 681  
TIBCServerProperties.LicenseInfo Property 676  
TIBCServerProperties.LicenseMaskInfo Property 676  
TIBCServerProperties.VersionInfo Property 676  
TIBCSQL Class 521  
TIBCSQL.BreakExec Method 530  
TIBCSQL.Connection Property 525  
TIBCSQL.CreateProcCall Method 530  
TIBCSQL.DescribeParams Property 526  
TIBCSQL.ExecuteNext Method 531  
TIBCSQL.FindParam Method 531  
TIBCSQL.Handle Property 526  
TIBCSQL.ParamByName Method 532  
TIBCSQL.Params Property 526  
TIBCSQL.SQLType Property 527  
TIBCSQL.Transaction Property 528  
TIBCSQLMonitor Class 807  
TIBCStatisticalService Class 681  
TIBCStatisticalService.Database Property 684  
TIBCStatisticalService.Options Property 684  
TIBCStatisticalService.TableNames Property 684  
TIBCSatOption Enumeration 712  
TIBCSatOptions Set 708  
TIBCStoredProc Class 533  
TIBCStoredProc.ExecProc Method 557  
TIBCStoredProc.LockMode Property 549  
TIBCStoredProc.Prepare Method 558  
TIBCStoredProc.PrepareSQL Method 558  
TIBCStoredProc.StoredProcName Property 549  
TIBCTable Class 559  
TIBCTable.FetchAll Property 575  
TIBCTable.LockMode Property 576  
TIBCTable.TableName Property 576  
TIBCTraceService Class 685  
TIBCTraceService.Config Property 688  
TIBCTraceService.ListTraceSessions Method 690  
TIBCTraceService.ResumeTrace Method 690  
TIBCTraceService.SessionName Property 688  
TIBCTraceService.StartTrace Method 690  
TIBCTraceService.StopTrace Method 691  
TIBCTraceService.SuspendTrace Method 691  
TIBCTransaction Class 576  
TIBCTransaction.Active Property 580  
TIBCTransaction.AddConnection Method 584  
TIBCTransaction.Commit Method 585  
TIBCTransaction.CommitRetaining Method 585  
TIBCTransaction.Connections Property(Indexer) 580  
TIBCTransaction.ConnectionsCount Property 581  
TIBCTransaction.DefaultCloseAction Property 581  
TIBCTransaction.DefaultConnection Property 582  
TIBCTransaction.FindDefaultConnection Method 585  
TIBCTransaction.Handle Property 582  
TIBCTransaction.IsolationLevel Property 582  
TIBCTransaction.OnError Event 589  
TIBCTransaction.Params Property 583  
TIBCTransaction.ReleaseSavepoint Method 586  
TIBCTransaction.RemoveConnection Method 586

TIBCTransaction.Rollback Method	587	TLoaderProgressEvent Procedure Reference	132
TIBCTransaction.RollbackRetaining Method	587	TLocateExOption Enumeration	844
TIBCTransaction.RollbackSavepoint Method	587	TLocateExOptions Set	841
TIBCTransaction.StartSavepoint Method	588	TLockMode Enumeration	358
TIBCTransactionAction Enumeration	593	TMacro Class	342
TIBCTransactionAdvise Enumeration	712	TMacro.Active Property	343
TIBCTransactionErrorEvent Procedure Reference	592	TMacro.AsDateTime Property	344
TIBCTransactionGlobalAction Enumeration	713	TMacro.AsFloat Property	344
TIBCTransactionState Enumeration	713	TMacro.AsInteger Property	344
TIBCUpsideSQL Class	590	TMacro.AsString Property	345
TIBCUserInfo Class	692	TMacro.Name Property	345
TIBCUserInfo.ActiveUser Property	693	TMacro.Value Property	345
TIBCUserInfo.DefaultRole Property	694	TMacros Class	346
TIBCUserInfo.Description Property	694	TMacros.AssignValues Method	348
TIBCUserInfo.FirstName Property	694	TMacros.FindMacro Method	348
TIBCUserInfo.GroupID Property	694	TMacros.IsEqual Method	349
TIBCUserInfo.GroupName Property	695	TMacros.Items Property(Indexer)	347
TIBCUserInfo.LastName Property	695	TMacros.MacroByName Method	349
TIBCUserInfo.MiddleName Property	695	TMacros.Scan Method	350
TIBCUserInfo.Password Property	696	TMapRule Class	85
TIBCUserInfo.SQLRole Property	696	TMapRule.DBLengthMax Property	87
TIBCUserInfo.SystemUserName Property	696	TMapRule.DBLengthMin Property	87
TIBCUserInfo.UserID Property	697	TMapRule.DBScaleMax Property	87
TIBCUserInfo.UserName Property	697	TMapRule.DBScaleMin Property	87
TIBCValidateOption Enumeration	713	TMapRule.DBType Property	88
TIBCValidateOptions Set	708	TMapRule.FieldLength Property	88
TIBCValidationService Class	697	TMapRule.FieldName Property	88
TIBCValidationService.Database Property	700	TMapRule.FieldScale Property	88
TIBCValidationService.FetchLimboTransactionInfo Method	704	TMapRule.IgnoreErrors Property	89
TIBCValidationService.FixLimboTransactionErrors Method	704	TMemDataSet Class	847
TIBCValidationService.GlobalAction Property	701	TMemDataSet.ApplyUpdates Method	855
TIBCValidationService.LimboTransactionInfo Property(Indexer)	701	TMemDataSet.CachedUpdates Property	850
TIBCValidationService.LimboTransactionInfoCount Property	701	TMemDataSet.CancelUpdates Method	856
TIBCValidationService.Options Property	702	TMemDataSet.CommitUpdates Method	857
TIBCValidationService.RecoverTwoPhaseGlobal Property	702	TMemDataSet.DeferredPost Method	858
TIBCValidationService.SweepDatabase Method	704	TMemDataSet.GetBlob Method	858
TIBCVersionInfo Class	705	TMemDataSet.IndexFieldNames Property	851
TIBCVersionInfo.ServerImplementation Property	706	TMemDataSet.LocalConstraints Property	851
TIBCVersionInfo.ServerVersion Property	706	TMemDataSet.LocalUpdate Property	852
TIBCVersionInfo.ServiceVersion Property	706	TMemDataSet.Locate Method	860
TimestampName Property	643	TMemDataSet.LocateEx Method	862
TLabelSet Enumeration	357	TMemDataSet.OnUpdateError Event	868
		TMemDataSet.OnUpdateRecord Event	869
		TMemDataSet.Prepare Method	863
		TMemDataSet.Prepared Property	852
		TMemDataSet.RestoreUpdates Method	864
		TMemDataSet.RevertRecord Method	864
		TMemDataSet.SaveToXML Method	865



- TMemDataSet.UnPrepare Method 866
  - TMemDataSet.UpdateRecordTypes Property 852
  - TMemDataSet.UpdateResult Method 866
  - TMemDataSet.UpdatesPending Property 853
  - TMemDataSet.UpdateStatus Method 867
  - TMonitorOption Enumeration 167
  - TMonitorOptions Set 166
  - TObjectType Class 833
  - TObjectType.AttributeByName Method 837
  - TObjectType.AttributeCount Property 835
  - TObjectType.Attributes Property(Indexer) 835
  - TObjectType.DataType Property 836
  - TObjectType.FindAttribute Method 838
  - TObjectType.Size Property 836
  - TOnErrorEvent Procedure Reference 156
  - TOnSQLEvent Procedure Reference 166
  - TPoolingOptions Class 350
  - TPoolingOptions.ConnectionLifetime Property 352
  - TPoolingOptions.MaxPoolSize Property 352
  - TPoolingOptions.MinPoolSize Property 352
  - TPoolingOptions.Validate Property 353
  - TraceFlags Property 161
  - Transaction Property
    - TCustomIBCDataSet 393
    - TIBCAlerter 718
    - TIBCConfigService 621
    - TIBCMetaData 493
    - TIBCScript 806
    - TIBCSQL 528
  - TransactionCount Property 461
  - Transactions Property(Indexer) 462
  - TRefreshOption Enumeration 358
  - TRefreshOptions Set 356
  - TRetryMode Enumeration 359
  - TrHandle Property
    - TCustomIBCArray 733
    - TIBCBlob 784
  - TrimFixedChar Property 294
  - Truncate Method 827
  - tsCommitState 713
  - TSharedObject Class 838
  - TSharedObject.AddRef Method 840
  - TSharedObject.RefCount Property 840
  - TSharedObject.Release Method 841
  - tsLimboState 713
  - TSortType Enumeration 845
  - tsRollbackState 713
  - tsUnknownState 713
  - TUpdateExecuteEvent Procedure Reference 356
  - TUpdateRecKind Enumeration 846
  - TUpdateRecKinds Set 842
  - TVirtualTable Class 872
  - TVirtualTable.AddField Method 880
  - TVirtualTable.Assign Method 881
  - TVirtualTable.Clear Method 881
  - TVirtualTable.DeleteField Method 882
  - TVirtualTable.DeleteFields Method 882
  - TVirtualTable.LoadFromFile Method 882
  - TVirtualTable.LoadFromStream Method 883
  - TVirtualTable.Options Property 877
  - TVirtualTable.SaveToFile Method 883
  - TVirtualTable.SaveToStream Method 884
  - TVirtualTableOption Enumeration 885
  - TVirtualTableOptions Set 885
- ## - U -
- ukDelete 846
  - ukInsert 846
  - ukUpdate 846
  - Unicode Character Data 48
  - UniDirectional Property 231
  - UnLock Method 250
  - UnPrepare Method
    - TCustomDASQL 269
    - TMemDataSet 866
  - Update Method 363
  - UpdateAllFields Property 294
  - UpdateBatchSize Property 294
  - UpdateCount Property 770
  - UpdateRecordTypes Property 852
  - UpdateResult Method 866
  - UpdatesPending Property 853
  - UpdateStatus Method 867
  - UpdateTransaction Property 394
  - Updating Data with IBDAC Dataset Components 36
  - UpdatingTable Property 520
  - UseDefConnection Variable 595
  - UseNBackup Property 609
  - UserDatabase Property 667
  - UserID Property 697
  - UserInfo Property 667
  - UserInfos Property(Indexer) 667
  - UserInfosCount Property 668
  - Username Property

Username Property		WriteArray Method	747
TCustomDACConnection	189	WriteArraySlice Method	748
TIBConnection	462	WriteBlob Method	787
TIBUserInfo	697	Writes Property	774
UsernameLabel Property	180	Writing GUI Applications with IBDAC	58
UserNames Property	770		
UseUnicode Property	474		
Using Several DAC Products in One IDE	57		

## - V -

Validate Property	353
Value Property	
TDAParam	330
TMacro	345
Verbose Property	
TIBCBBackupRestoreService	610
TIBCBBackupService	614
TIBCRestoreService	663
Version Property	771
VersionInfo Property	676
VirtualTable Unit Members	871
voCheckDB	713
voIgnoreChecksum	713
voKillShadows	713
voMendDB	713
voPersistentData	885
voStored	885
voValidateDB	713
voValidateFull	713

## - W -

WaitExecuting Method	270
WALAveragGroupCommitSize Property	771
WALAveragelIOSize Property	771
WALBufferSize Property	772
WALCheckpointLength Property	772
WALCurCheckpointInterval Property	772
WALGroupCommitWaitUSecs Property	773
WALNumCommits Property	773
WALNumIO Property	773
WALPrvCheckpointFilename Property	774
WALPrvCheckpointPartOffset Property	774
What's New	9
Working in an Unstable Network	51
Write Method	828