

# Table of Contents

|  |    |
|--|----|
| <b>Overview</b>                                    | 1  |
| <b>Editions</b>                                    | 3  |
| <b>Getting Started</b>                             | 5  |
| <b>Features</b>                                    | 9  |
| <b>What's New</b>                                  | 12 |
| <b>Demo Projects</b>                               | 18 |
| <b>Component List</b>                              | 23 |
| <b>Hierarchy Chart</b>                             | 25 |
| <b>Requirements</b>                                | 26 |
| <b>Compatibility</b>                               | 27 |
| <b>Installation</b>                                | 29 |
| <b>Deployment</b>                                  | 32 |
| <b>Licensing and Subscriptions</b>                 | 33 |
| <b>Getting Support</b>                             | 34 |
| <b>Frequently Asked Questions</b>                  | 35 |
| <b>Using MyDAC</b>                                 | 40 |
| <b>Updating Data with MyDAC Dataset Components</b> | 40 |
| <b>Master/Detail Relationships</b>                 | 41 |
| <b>Migration from BDE</b>                          | 43 |
| <b>Secure Connections</b>                          | 45 |
| <b>Network Tunneling</b>                           | 47 |
| <b>Embedded Server</b>                             | 49 |
| <b>National Characters</b>                         | 51 |
| <b>Working in an Unstable Network</b>              | 52 |
| <b>Disconnected Mode</b>                           | 53 |
| <b>Data Type Mapping</b>                           | 54 |
| <b>Data Encryption</b>                             | 58 |
| <b>Increasing Performance</b>                      | 60 |
| <b>Connection Pooling</b>                          | 62 |

|   |     |
|---|-----|
| <b>Macros</b>   | 64  |
| <b>Using Several DAC Products in One IDE</b>              | 65  |
| <b>DataSet Manager</b>                                    | 66  |
| <b>DBMonitor</b>  | 71  |
| <b>Migration Wizard</b>                                   | 72  |
| <b>Writing GUI Applications with MyDAC</b>                | 73  |
| <b>Compatibility with Previous Versions</b>               | 74  |
| <b>dbForge Fusion for MySQL</b>                           | 75  |
| <b>MyBuilder Add-In</b>                                   | 80  |
| <b>64-bit Development with Embarcadero RAD Studio XE2</b> | 81  |
| <b>Database Specific Aspects of 64-bit Development</b>    | 85  |
| <b>Reference</b>  | 86  |
| <b>CRAccess</b>   | 88  |
| Classes   | 89  |
| .TCRCursor Class  | 89  |
| Members   | 89  |
| Types   | 90  |
| .TBeforeFetchProc Procedure Reference                     | 90  |
| Enumerations  | 91  |
| .TCRIsoLevel Enumeration                                  | 91  |
| .TCRTransactionAction Enumeration                         | 91  |
| <b>CRBatchMove</b>  | 92  |
| Classes   | 93  |
| .TCRBatchMove Class                                       | 93  |
| Members   | 93  |
| Properties  | 94  |
| Methods   | 99  |
| Events  | 99  |
| Types   | 101 |
| ...TCRBatchMoveProgressEvent Procedure Reference          | 101 |
| Enumerations  | 102 |
| ...TCRBatchMode Enumeration                               | 102 |
| ...TCRFieldMappingMode Enumeration                        | 102 |
| <b>CRDataTypeMap</b>                                      | 103 |
| Classes   | 104 |
| ...EDataMappingError Class                                | 104 |
| Members   | 104 |
| ...EDataTypeMappingError Class                            | 104 |
| Members   | 105 |
| ...EInvalidDBTypeMapping Class                            | 105 |
| Members   | 105 |
| ...EInvalidFieldTypeMapping Class                         | 105 |
| Members   | 106 |
| ...EUnsupportedDataTypeMapping Class                      | 106 |
| Members   | 106 |
| ...TMapRule Class   | 106 |
| Members   | 106 |

|   |     |
|---|-----|
| Properties.....                                     | 107 |
| <b>CREncryption</b> .....                           | 110 |
| Classes .....                                       | 111 |
| ...TCREncryptor Class.....                          | 111 |
| Members.....  | 111 |
| Properties.....                                     | 111 |
| Methods.....  | 113 |
| Enumerations.....                                   | 115 |
| ...TCREncDataHeader Enumeration.....                | 115 |
| ...TCREncryptionAlgorithm Enumeration.....          | 115 |
| ...TCRHashAlgorithm Enumeration.....                | 116 |
| ...TCRInvalidHashAction Enumeration.....            | 116 |
| <b>CRVio</b> .....                                  | 117 |
| Classes .....                                       | 118 |
| ...THttpOptions Class.....                          | 118 |
| Members.....  | 118 |
| Properties.....                                     | 118 |
| ...TProxyOptions Class.....                         | 120 |
| Members.....  | 120 |
| Properties.....                                     | 120 |
| <b>DADump</b> .....                                 | 123 |
| Classes .....                                       | 124 |
| ...TDADump Class.....                               | 124 |
| Members.....  | 124 |
| Properties.....                                     | 125 |
| Methods.....  | 127 |
| Events.....   | 130 |
| ...TDADumpOptions Class.....                        | 132 |
| Members.....  | 132 |
| Properties.....                                     | 132 |
| Types .....   | 134 |
| ...TDABackupProgressEvent Procedure Reference.....  | 134 |
| ...TDARestoreProgressEvent Procedure Reference..... | 134 |
| <b>DALoader</b> .....                               | 135 |
| Classes .....                                       | 136 |
| ...TDAColumn Class.....                             | 136 |
| Members.....  | 136 |
| Properties.....                                     | 136 |
| ...TDAColumns Class.....                            | 137 |
| Members.....  | 138 |
| Properties.....                                     | 138 |
| ...TDALoader Class.....                             | 138 |
| Members.....  | 139 |
| Properties.....                                     | 139 |
| Methods.....  | 141 |
| Events.....   | 143 |
| Types .....   | 146 |
| ...TDAPutDataEvent Procedure Reference.....         | 146 |
| ...TGetColumnDataEvent Procedure Reference.....     | 146 |
| ...TLoaderProgressEvent Procedure Reference.....    | 147 |
| <b>DAScript</b> .....                               | 148 |
| Classes .....                                       | 149 |

|  |     |
|--|-----|
| ...TDAScript Class.....                                  | 149 |
| Members.....   | 149 |
| Properties.....  | 150 |
| Methods.....   | 155 |
| Events.....  | 158 |
| ...TDASTatement Class.....                               | 159 |
| Members.....   | 160 |
| Properties.....  | 160 |
| ...TDASTatements Class.....                              | 163 |
| Members.....   | 164 |
| Properties.....  | 164 |
| Types .....  | 165 |
| ...TAfterStatementExecuteEvent Procedure Reference.....  | 165 |
| ...TBeforeStatementExecuteEvent Procedure Reference..... | 165 |
| ...TOnErrorEvent Procedure Reference.....                | 165 |
| Enumerations.....  | 167 |
| ...TErrorAction Enumeration.....                         | 167 |
| <b>DASQLMonitor</b> .....                                | 168 |
| Classes .....  | 169 |
| ...TCustomDASQLMonitor Class.....                        | 169 |
| Members.....   | 169 |
| Properties.....  | 170 |
| Events.....  | 171 |
| ...TDBMonitorOptions Class.....                          | 172 |
| Members.....   | 172 |
| Properties.....  | 172 |
| Types .....  | 174 |
| ...TDATraceFlags Set.....                                | 174 |
| ...TMonitorOptions Set.....                              | 174 |
| ...TOnSQLEvent Procedure Reference.....                  | 174 |
| Enumerations.....  | 175 |
| ...TDATraceFlag Enumeration.....                         | 175 |
| ...TMonitorOption Enumeration.....                       | 175 |
| <b>DBAccess</b> .....                                    | 177 |
| Classes .....  | 180 |
| ...EDAError Class.....                                   | 181 |
| Members.....   | 181 |
| Properties.....  | 181 |
| ...TCRDataSource Class.....                              | 182 |
| Members.....   | 182 |
| ...TCustomConnectDialog Class.....                       | 182 |
| Members.....   | 183 |
| Properties.....  | 183 |
| Methods.....   | 187 |
| ...TCustomDAConnection Class.....                        | 188 |
| Members.....   | 188 |
| Properties.....  | 189 |
| Methods.....   | 194 |
| Events.....  | 203 |
| ...TCustomDADataset Class.....                           | 204 |
| Members.....   | 204 |
| Properties.....  | 209 |
| Methods.....   | 224 |
| Events.....  | 238 |

|  |     |
|--|-----|
| ...TCustomDASQL Class.....                     | 241 |
| Members.....                                   | 241 |
| Properties.....                                | 242 |
| Methods.....                                   | 248 |
| Events.....                                    | 252 |
| ...TCustomDAUpdateSQL Class.....               | 253 |
| Members.....                                   | 253 |
| Properties.....                                | 254 |
| Methods.....                                   | 258 |
| ...TDAConnectionOptions Class.....             | 260 |
| Members.....                                   | 260 |
| Properties.....                                | 260 |
| ...TDADatasetOptions Class.....                | 262 |
| Members.....                                   | 262 |
| Properties.....                                | 263 |
| ...TDAEncryptionOptions Class.....             | 269 |
| Members.....                                   | 270 |
| Properties.....                                | 270 |
| ...TDAMapRule Class.....                       | 271 |
| Members.....                                   | 271 |
| Properties.....                                | 272 |
| ...TDAMapRules Class.....                      | 274 |
| Members.....                                   | 275 |
| Methods.....                                   | 275 |
| ...TDAMetaData Class.....                      | 282 |
| Members.....                                   | 283 |
| Properties.....                                | 284 |
| Methods.....                                   | 287 |
| ...TDAParam Class.....                         | 289 |
| Members.....                                   | 290 |
| Properties.....                                | 290 |
| Methods.....                                   | 295 |
| ...TDAParams Class.....                        | 298 |
| Members.....                                   | 298 |
| Properties.....                                | 298 |
| Methods.....                                   | 299 |
| ...TDATransaction Class.....                   | 300 |
| Members.....                                   | 300 |
| Properties.....                                | 301 |
| Methods.....                                   | 302 |
| Events.....                                    | 303 |
| ...TMacro Class.....                           | 304 |
| Members.....                                   | 304 |
| Properties.....                                | 304 |
| ...TMacros Class.....                          | 307 |
| Members.....                                   | 307 |
| Properties.....                                | 307 |
| Methods.....                                   | 308 |
| ...TPoolingOptions Class.....                  | 310 |
| Members.....                                   | 310 |
| Properties.....                                | 311 |
| Types.....                                     | 313 |
| ...TAfterExecuteEvent Procedure Reference..... | 313 |
| ...TAfterFetchEvent Procedure Reference.....   | 313 |
| ...TBeforeFetchEvent Procedure Reference.....  | 314 |

|  |            |
|--|------------|
| ...TConnectionLostEvent Procedure Reference.....     | 314        |
| ...TDAConnectionErrorEvent Procedure Reference.....  | 314        |
| ...TDATransactionErrorEvent Procedure Reference..... | 315        |
| ...TRefreshOptions Set.....                          | 315        |
| ...TUpdateExecuteEvent Procedure Reference.....      | 315        |
| <b>Enumerations.....</b>                             | <b>317</b> |
| ...TLabelSet Enumeration.....                        | 317        |
| ...TLockMode Enumeration.....                        | 317        |
| ...TRefreshOption Enumeration.....                   | 318        |
| ...TRetryMode Enumeration.....                       | 318        |
| <b>Variables.....</b>                                | <b>319</b> |
| ...BaseSQLOldBehavior Variable.....                  | 319        |
| ...ChangeCursor Variable.....                        | 319        |
| ...MacroChar Variable.....                           | 320        |
| ...SQLGeneratorCompatibility Variable.....           | 320        |
| <b>Devart.Dac.DataAdapter.....</b>                   | <b>321</b> |
| <b>Classes.....</b>                                  | <b>322</b> |
| ...DADDataAdapter Class.....                         | 322        |
| Members.....   | 322        |
| Properties.....                                      | 323        |
| Methods.....   | 323        |
| <b>Devart.MyDac.DataAdapter.....</b>                 | <b>325</b> |
| <b>Classes.....</b>                                  | <b>326</b> |
| ...MyDataAdapter Class.....                          | 326        |
| Members.....   | 326        |
| <b>MemData.....</b>                                  | <b>328</b> |
| <b>Classes.....</b>                                  | <b>329</b> |
| ...TAttribute Class.....                             | 329        |
| Members.....   | 329        |
| Properties.....                                      | 330        |
| ...TBlob Class.....                                  | 333        |
| Members.....   | 333        |
| Properties.....                                      | 334        |
| Methods.....   | 335        |
| ...TCompressedBlob Class.....                        | 340        |
| Members.....   | 340        |
| ...TDBObject Class.....                              | 341        |
| Members.....   | 342        |
| ...TObjectType Class.....                            | 342        |
| Members.....   | 342        |
| Properties.....                                      | 343        |
| Methods.....   | 344        |
| ...TSharedObject Class.....                          | 346        |
| Members.....   | 346        |
| Properties.....                                      | 346        |
| Methods.....   | 347        |
| <b>Types.....</b>                                    | <b>349</b> |
| ...TLocateExOptions Set.....                         | 349        |
| ...TUpdateRecKinds Set.....                          | 349        |
| <b>Enumerations.....</b>                             | <b>350</b> |
| ...TConnLostCause Enumeration.....                   | 350        |
| ...TDANumericType Enumeration.....                   | 351        |
| ...TLocateExOption Enumeration.....                  | 351        |

|  |            |
|--|------------|
| ...TSortType Enumeration.....                    | 351        |
| ...TUpdateRecKind Enumeration.....               | 352        |
| <b>MemDS</b> .....                               | <b>353</b> |
| Classes .....                                    | <b>354</b> |
| ...TMemDataSet Class .....                       | 354        |
| Members.....                                     | 354        |
| Properties.....                                  | 355        |
| Methods .....                                    | 359        |
| Events.....                                      | 369        |
| Variables .....                                  | <b>372</b> |
| ...DoNotRaiseExcetionOnUaFail Variable.....      | 372        |
| ...SendDataSetChangeEventAfterOpen Variable..... | 372        |
| <b>MyAccess</b> .....                            | <b>373</b> |
| Classes .....                                    | <b>375</b> |
| ...TCustomMyConnection Class .....               | 376        |
| Members.....                                     | 376        |
| Properties.....                                  | 378        |
| Methods .....                                    | 382        |
| ...TCustomMyConnectionOptions Class.....         | 388        |
| Members.....                                     | 388        |
| Properties.....                                  | 389        |
| ...TCustomMyDataSet Class .....                  | 391        |
| Members.....                                     | 392        |
| Properties.....                                  | 396        |
| Methods .....                                    | 404        |
| ...TCustomMyStoredProc Class .....               | 411        |
| Members.....                                     | 411        |
| Properties.....                                  | 416        |
| Methods .....                                    | 421        |
| ...TCustomMyTable Class .....                    | 427        |
| Members.....                                     | 427        |
| Properties.....                                  | 432        |
| Methods .....                                    | 438        |
| ...TMyCommand Class.....                         | 443        |
| Members.....                                     | 443        |
| Properties.....                                  | 445        |
| Methods .....                                    | 447        |
| ...TMyConnection Class.....                      | 449        |
| Members.....                                     | 449        |
| Properties.....                                  | 451        |
| ...TMyConnectionOptions Class .....              | 455        |
| Members.....                                     | 456        |
| Properties.....                                  | 457        |
| ...TMyConnectionSSLOptions Class.....            | 459        |
| Members.....                                     | 460        |
| Properties.....                                  | 460        |
| ...TMyDataSetOptions Class .....                 | 461        |
| Members.....                                     | 461        |
| Properties.....                                  | 463        |
| ...TMyDataSource Class.....                      | 470        |
| Members.....                                     | 471        |
| ...TMyEncryptor Class .....                      | 471        |
| Members.....                                     | 471        |
| ...TMyMetaData Class.....                        | 472        |

|   |            |
|---|------------|
| Members.....                                      | 472        |
| ...TMyQuery Class.....                            | 474        |
| Members.....                                      | 474        |
| Properties.....                                   | 479        |
| ...TMyStoredProc Class.....                       | 485        |
| Members.....                                      | 486        |
| Properties.....                                   | 490        |
| ...TMyTable Class.....                            | 496        |
| Members.....                                      | 497        |
| Properties.....                                   | 501        |
| ...TMyTableOptions Class.....                     | 508        |
| Members.....                                      | 508        |
| Properties.....                                   | 510        |
| ...TMyTransaction Class.....                      | 513        |
| Members.....                                      | 513        |
| ...TMyUpdateSQL Class.....                        | 513        |
| Members.....                                      | 514        |
| Types .....                                       | 515        |
| ...TMyUpdateExecuteEvent Procedure Reference..... | 515        |
| Enumerations.....                                 | 516        |
| ...TLockRecordType Enumeration.....               | 516        |
| ...TLockType Enumeration.....                     | 516        |
| ...TMyIsolationLevel Enumeration.....             | 517        |
| Routines .....                                    | 518        |
| ...GetServerList Procedure.....                   | 518        |
| Constants.....                                    | 519        |
| ...MydacVersion Constant.....                     | 519        |
| <b>MyBackup .....</b>                             | <b>520</b> |
| Classes .....                                     | 521        |
| ...TMyBackup Class.....                           | 521        |
| Members.....                                      | 521        |
| Properties.....                                   | 522        |
| Methods.....                                      | 528        |
| Events.....                                       | 529        |
| Types .....                                       | 531        |
| ...TMyTableMsgEvent Procedure Reference.....      | 531        |
| Enumerations.....                                 | 532        |
| ...TMyBackupMode Enumeration.....                 | 532        |
| ...TMyBackupPriority Enumeration.....             | 532        |
| ...TMyRestoreDuplicates Enumeration.....          | 533        |
| <b>MyBuilderClient .....</b>                      | <b>534</b> |
| Classes .....                                     | 535        |
| ...TMyBuilder Class.....                          | 535        |
| Members.....                                      | 535        |
| Properties.....                                   | 536        |
| Methods.....                                      | 537        |
| <b>MyClasses .....</b>                            | <b>539</b> |
| Classes .....                                     | 540        |
| ...EMyError Class.....                            | 540        |
| Members.....                                      | 540        |
| Properties.....                                   | 540        |
| Enumerations.....                                 | 542        |
| ...TMyProtocol Enumeration.....                   | 542        |



|  |     |
|--|-----|
| Variables .....                        | 543 |
| ..._Strings65535ToMemo Variable .....  | 543 |
| <b>MyConnectionPool</b> .....          | 544 |
| Classes .....                          | 545 |
| ...TMyConnectionPoolManager Class..... | 545 |
| Members.....                           | 545 |
| <b>MyDacVcl</b> .....                  | 546 |
| Classes .....                          | 547 |
| ...TMyConnectDialog Class.....         | 547 |
| Members.....                           | 547 |
| Properties.....                        | 548 |
| <b>MyDump</b> .....                    | 552 |
| Classes .....                          | 553 |
| ...TMyDump Class.....                  | 553 |
| Members.....                           | 553 |
| Properties.....                        | 554 |
| ...TMyDumpOptions Class.....           | 557 |
| Members.....                           | 558 |
| Properties.....                        | 558 |
| Types .....                            | 561 |
| ...TMyDumpObjects Set.....             | 561 |
| Enumerations.....                      | 562 |
| ...TMyDumpObject Enumeration.....      | 562 |
| <b>MyEmbConnection</b> .....           | 563 |
| Classes .....                          | 564 |
| ...TMyEmbConnection Class.....         | 564 |
| Members.....                           | 564 |
| Properties.....                        | 566 |
| Events.....                            | 570 |
| <b>MyLoader</b> .....                  | 573 |
| Classes .....                          | 574 |
| ...TMyColumn Class.....                | 574 |
| Members.....                           | 574 |
| ...TMyLoader Class.....                | 574 |
| Members.....                           | 575 |
| Properties.....                        | 576 |
| Types .....                            | 578 |
| ...TMyLoaderOptions Set.....           | 578 |
| Enumerations.....                      | 579 |
| ...TMyDuplicateKeys Enumeration.....   | 579 |
| ...TMyLoaderOption Enumeration.....    | 579 |
| <b>MyScript</b> .....                  | 580 |
| Classes .....                          | 581 |
| ...TMyScript Class.....                | 581 |
| Members.....                           | 581 |
| Properties.....                        | 582 |
| <b>MyServerControl</b> .....           | 585 |
| Classes .....                          | 586 |
| ...TMyServerControl Class.....         | 586 |
| Members.....                           | 586 |
| Properties.....                        | 592 |

|   |     |
|---|-----|
| Methods .....                           | 598 |
| <b>MySqlApi</b> .....                   | 610 |
| Types .....                             | 611 |
| ...TMyLogEvent Procedure Reference..... | 611 |
| Variables .....                         | 612 |
| ...MySQLClientLibrary Variable.....     | 612 |
| <b>MySQLMonitor</b> .....               | 613 |
| Classes .....                           | 614 |
| ...TMySQLMonitor Class.....             | 614 |
| Members.....                            | 614 |
| <b>VirtualTable</b> .....               | 615 |
| Classes .....                           | 616 |
| ...TVirtualTable Class.....             | 616 |
| Members.....                            | 616 |
| Properties.....                         | 618 |
| Methods .....                           | 619 |
| Types .....                             | 626 |
| ...TVirtualTableOptions Set.....        | 626 |
| Enumerations.....                       | 627 |
| ...TVirtualTableOption Enumeration..... | 627 |

# 1 Overview

Data Access Components for MySQL (MyDAC) is a library of components that provides direct access to MySQL database servers from Delphi, Delphi for .NET, C++Builder, Free Pascal, and Kylix. MyDAC can connect directly to MySQL server or work through the MySQL client library. The MyDAC library is designed to help programmers develop faster and cleaner MySQL database applications. MyDAC is a complete replacement for standard MySQL connectivity solutions and presents an efficient alternative to the Borland Database Engine for access to MySQL.

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

## Advantages of MyDAC Technology

MyDAC is a direct database connectivity wrapper built specifically for the MySQL server. MyDAC offers wide coverage of the MySQL feature set, supports both client and direct connection modes, and emphasizes optimized data access strategies.

### Wide Coverage of MySQL Features

By providing access to the most advanced database functionality, MyDAC allows developers to harness the full capabilities of the MySQL server and optimize their database applications. MyDAC provides a complete support of MySQL Embedded Server, row-level locking, using HANDLER statements, MySQL administration tasks. Get a full list of supported MySQL features in the [Features](#) topic.

### Native Connection Options

MyDAC offers two connection modes to the MySQL server: Direct connection and connection through the standard MySQL Client in Client mode. MyDAC-based database applications are easy to deploy, do not require installation of other data provider layers (such as BDE), and tend to be faster than those that use standard data connectivity solutions. See the [How does MyDAC work](#) section.

### Optimized Code

The goal of MyDAC is to enable developers to write efficient and flexible database applications. The MyDAC 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 MyDAC to optimize your database applications in [Increasing Performance](#).

### Compatibility with other Connectivity Methods

The MyDAC interface retains compatibility with standard VCL data access components BDE. Existing BDE-based applications can be easily migrated to MyDAC and enhanced to take advantage of MySQL-specific features. Migration of a Delphi project can be automated with the BDE Migration Wizard. Find out more about Migration Wizard in the [Migration from BDE](#) topic.

### Development and Support

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

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

## Key Features

- [Direct](#) access to server data without using client library. Does not require installation of other data provider layers (such as BDE and ODBC)
- VCL, VCL.NET, and CLX versions of library available
- Full support of the [latest versions of MySQL Server](#)
- Support for all MySQL Server data types
- [Disconnected Model](#) with automatic connection control for working with data offline
- [Local Failover](#) for detecting connection loss and implicitly reexecuting certain operations
- All types of local [sorting](#) and filtering, including by calculated and lookup fields
- [Automatic data updating](#) with [TMyQuery](#), [TMyTable](#), and [TMyStoredProc](#) components
- [Unicode](#) and [national charset](#) support
- Supports many MySQL-specific features, such as [locking](#), [SET](#) and ENUM types

- Advanced script execution functionality with [TMyScript](#) component
- Support for [using macros](#) in SQL
- Integration with [dbForge Fusion for MySQL](#) Standard Edition for performing advanced database development and administration tasks
- Easy migration from [BDE](#) with [Migration Wizard](#)
- Lets you use Professional Edition of [Delphi and C++Builder](#) to develop client/server applications
- Included annual [MyDAC Subscription](#) with [Priority Support](#)
- Licensed royalty-free per developer, per team, or per site

The full list of MyDAC features are available in the [Features](#) topic.

---

## 2 Editions

Data Access Components for MySQL comes in four basic editions levels: MyDAC Standard Edition, MyDAC Professional Edition, MyDAC Developer Edition, and MyDAC Trial Edition.

MyDAC Standard Edition includes the MyDAC basic connectivity components and the MyDAC Migration Wizard. MyDAC Standard Edition is a good choice for beginning MySQL developers and a cost-effective solution for database application developers who only need basic connectivity functionality for MySQL. MyDAC Professional Edition shows off the full power of MyDAC, enhancing MyDAC Standard Edition with support for MySQL-specific functionality, and some advanced connection management features. MyDAC Professional Edition is intended for serious application developers who want to take advantage of all the MySQL-specific functionality support provided by MyDAC.

MyDAC Developer Edition is a bundle package of MyDAC Professional Edition with dbForgeFusion for MySQL Standard Edition, an advanced add-in for MySQL database development and administration.

MyDAC Developer Edition is the best choice for enterprises and database industry professionals.

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

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

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

**MyDAC Edition Matrix**

| Feature   | Developer* | Professional** | Standard | Trial |
|---|------------|----------------|----------|-------|
| <b>Base Components</b>  |            |                |          |       |
| <a href="#">TMyConnection</a>   |            |                |          |       |
| <a href="#">TMyQuery</a>  |            |                |          |       |
| <a href="#">TMyCommand</a>  |            |                |          |       |
| <a href="#">TMyTable</a>  |            |                |          |       |
| <a href="#">TMyStoredProc</a>   | +          | +              | +        | +     |
| <a href="#">TMyUpdateSQL</a>  |            |                |          |       |
| <a href="#">TMyConnectDialog</a>  |            |                |          |       |
| <a href="#">TMySQLMonitor</a>   |            |                |          |       |
| <a href="#">TMyScript</a>   |            |                |          |       |
| <a href="#">TMyDataSource</a>   |            |                |          |       |
| <a href="#">TVirtualTable</a>   |            |                |          |       |
| <a href="#">TCRDBGrid</a>   |            |                |          |       |
| <a href="#">MyDataAdapter</a>   |            |                |          |       |
| <b>Additional Components</b>  |            |                |          |       |
| <a href="#">TMyEncryptor</a>  |            |                |          |       |
| <a href="#">TMyLoader</a>   |            |                |          |       |
| <a href="#">TMyDump</a>   |            |                |          |       |
| <a href="#">TMyBackup</a>   | +          | +              | -        | +     |
| <a href="#">TMyServerControl</a>  |            |                |          |       |
| <a href="#">TMyEmbConnection</a>  |            |                |          |       |
| <a href="#">TMyBuilder</a>  |            |                |          |       |
| <a href="#">TMyMetaData</a>   |            |                |          |       |
| <a href="#">TCRBatchMove</a>  |            |                |          |       |
| <b><a href="#">Direct connectivity</a></b> (without MySQL client)             | +          | +              | +        | +     |
| <b>Design-time features, including component editors and property editors</b> | +          | +              | +        | +     |
| <b><a href="#">DataSet Manager</a></b> ***                                    | +          | -              | -        | +     |

**Migration Wizard\*\*\***

+ + + +

**dbForge Fusion for MySQL Standard Edition\*\*\*\***

+ - - +

**Trial limitations\***

- - - +

\* Trial Edition is a fully working version of MyDAC Professional Edition for a trial period of 60 days on most supported IDEs. After the trial period expires you must either register or uninstall MyDAC. MyDAC Trial Edition for Kylix has an additional nag screen trial limitation. MyDAC 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.

\*\* Developer and Professional editions with source code are available. Migration Wizard, DataSet Manager source code, and code for other products, including dbForge Fusion for MySQL Standard Edition and SQL Builder for MySQL, is not distributed.

\*\*\* Not available for C++Builder, Delphi 8, FreePascal or Kylix.

\*\*\*\* List of environments this feature is compatible with you can find in the [Using dbForge Fusion for MySQL](#) topic

---

## 3 Getting Started

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

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

### What is MyDAC?

Data Access Components for MySQL (MyDAC) is a component library which provides direct connectivity to MySQL for Delphi, Delphi for .NET, C++-Builder, and Kylix, and helps you develop fast MySQL-based database applications with these environments.

Many MyDAC classes are based on VCL, VCL for .NET, and CLX classes and interfaces. MyDAC is a replacement for the [Borland Database Engine](#), it provides native database connectivity, and is specifically designed as an interface to the MySQL database.

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

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

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

### How does MyDAC work?

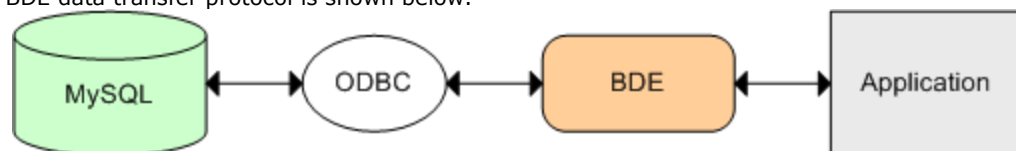
MyDAC allows you to connect to MySQL in two ways: in Client mode, using MySQL Client software, or in Direct mode. The chosen connection mode is regulated by the [Direct option](#).

In Direct mode, MyDAC connects to MySQL directly without using MySQL client software.

In Client mode, MyDAC connects to MySQL through the MySQL client library. MySQL client library is supplied with MySQL server.

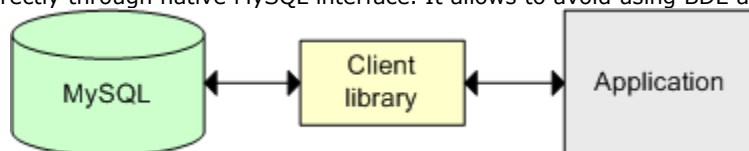
In comparison, the Borland Database Engine (BDE) uses several layers to access MySQL, and requires additional data access software to be installed on client machines.

The BDE data transfer protocol is shown below.



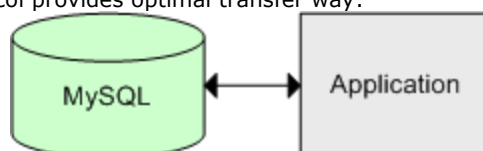
### BDE Connection Protocol

MyDAC works directly through native MySQL interface. It allows to avoid using BDE and ODBC:



### MyDAC Connection Flow Client Mode

Using MySQL network protocol provides optimal transfer way:



## MyDAC Connection Flow Direct Mode

### Installing MyDAC

To install MyDAC, complete the following steps.

1. Choose and download the version of the MyDAC installation program that is compatible with your IDE. For instance, if you are installing MyDAC 5.00, you should use the following files:

For BDS 2006 and Turbo - **mydac500d10\*.exe**

For Delphi 7 - **mydac500d7\*.exe**

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

2. Close all running Borland applications.
3. Launch the MyDAC installation program you downloaded in the first step and follow the instructions to install MyDAC.

By default, the MyDAC installation program should install compiled MyDAC libraries automatically on all IDEs except for Kylix. View the installation instructions for Kylix [here](#).

To check if MyDAC has been installed properly, launch your IDE and make sure that a MySQL Access page has been added to the Component palette and that a MySQL menu was added to the Menu bar. If you have bought MyDAC Professional Edition with Source Code or MyDAC Developer Edition with Source Code, you will be able to download both the compiled version of MyDAC and the MyDAC source code. The installation process for the compiled version is standard, as described above. The MyDAC source code must be compiled and installed manually. Consult the supplied *ReadmeSrc.txt* file for more details.

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

### Working with the MyDAC demo projects

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

To quickly get started working with MyDAC, launch and explore the introductory MyDAC demo project, *MyDacDemo*, from your IDE. This demo project is a collection of demos that show how MyDAC 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.

#### MyDacDemo Walkthrough

1. Launch your IDE.
2. Choose File → Open Project from the menu bar
3. Find the MyDAC directory and open the *MyDacDemo* project. This project should be located in the Demos\MyDacDemo folder.  
For example, if you are using Borland Developer Studio 2006, the demo project may be found at  
`\Program Files\Devart\MyDac for Delphi 2006\Demos\Win32\MyDacDemo\MyDacDemo.bdsproj`
4. Select Run → Run or press F9 to compile and launch the demo project. *MyDacDemo* should start, and a full-screen MyDAC Demo window with a toolbar, an explorer panel, and a view panel will open. The explorer panel will contain the list of all demo sub-projects included in *MyDacDemo*, 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 MySQL. However, you will not be able to actually retrieve data from MySQL or execute commands until you connect to the database.

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

Now you have a fully functional interface to your MySQL server. You will be able to go through the different demos, to browse tables, create and drop objects, and execute SQL 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 *MyDacDemo*. If some of these objects already exist in the database you have connected to, the following error message will appear.

*An error has occurred:*

*#42S01Table 'dept' already exists*

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



*%MyDAC% is the MyDAC 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. *MyDacDemo* will create the *MyDacDemo* objects on the server you have connected to.

7. Choose a demo that demonstrates an aspect of working with MySQL 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 MySQL tables, select the Table demo from the "Working with Components" folder. A simple MySQL 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 *MyDacDemo* toolbar to find out how the demo you selected was implemented. The source code behind the demo project will appear in the view panel. Try to find the places where MyDAC components are used to connect to the database.
9. Click on the "Form as text" button in the *MyDacDemo* toolbar to view the code behind the interface to the demo. Try to find the places where MyDAC 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 MyDAC components.

#### **General demos**

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

#### **MySQL-specific demos**

A collection of projects that demonstrate how to incorporate MySQL features in database applications.

11. When you are finished working with the project, click on the "Drop" button in the *MyDacDemo* toolbar to remove all the schema objects added in Step 6.

#### Other MyDAC demo projects

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

## **Compiling and deploying your MyDAC project**

#### Compiling MyDAC-based projects

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

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

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

#### Deploying MyDAC-based projects

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

- The MySQL client library, if connecting using MySQL client.
- The MyDAC bpl files, if compiling with runtime packages.
- The MyDAC assembly files, if are using VCL for .NET components.

If you are evaluating deploying projects with MyDAC 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 MyDAC Trial Edition for C++Builder will only work if the C++Builder IDE is launched. More information about MyDAC Trial Edition limitations is provided [here](#). A list of the files which may need to be deployed with MyDAC-based applications is included in the [Deployment](#) topic.

## **Using the MyDAC documentation**

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

The MyDAC documentation includes a detailed reference of all the MyDAC components and classes. Many of the MyDAC components and classes inherit or implement members from other VCL, VCL for .NET, CLX 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/CLX class an MyDAC component is inherited from, see the corresponding topic in your IDE documentation.

At install time, the MyDAC documentation is integrated into your IDE. It can be invoked from the MySQL 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 MyDAC

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

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

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

---

## 4 Features

### General usability:

- Direct access to server data without using client library. 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, VCL for .NET, and CLX 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
- Support for all existing MySQL protocols including the prepared statement (binary) protocol
- SSH and SSL [encrypted connection](#) support with [Devart SecureBridge](#)
- Full support for all current authentication protocols
- Ability to [search for installed MySQL servers in a local network](#)
- [Connection timeout](#) and [command timeout](#) management

### Compatibility:

- [Full support of the latest versions of MySQL](#)
- [Support for Embedded MySQL server](#)
- Support for all MySQL Server data types
- [Compatible with all IDE versions starting with Delphi 5, C++Builder 5, FreePascal, and Kylix 2 \(except Delphi 8\)](#)
- Includes provider for UniDAC Standard Edition
- [Wide reporting component support](#), including support for InfoPower, ReportBuilder, FastReport
- Wide support of all standard Borland and third-party visual data-aware controls
- Allows you to use Professional Edition of Delphi and C++Builder to develop client/server applications

### MySQL Server technology support:

- Fast record insertion with [TMyLoader](#) component
- [HANDLER syntax support](#)
- [Transaction isolation level support](#)
- Possibility to retrieve [last auto-incremented value](#)
- [Session identifier](#) retrieval
- [Server object information](#) retrieval
- [Row-level](#) and [table-level](#) locking support

### 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 [TMyQuery](#), [TMyTable](#) and [TMyStoredProc](#) components
- [Automatic record refreshing](#)
- [Automatic query preparing](#)

- [Automatic checking for row modifications](#) by another user
- 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 [TMyTable](#) component
- [BLOB compression](#) support
- Support for [using macros](#) in SQL
- [FmtBCD fields support](#)
- Ability to customize update commands by attaching external components to [TMyUpdateSQL](#) objects
- Ability to perform MySQL administration tasks with the [TMyServerControl](#) component
- [Value range retrieval for ENUM and SET fields](#)
- Retrieval of output parameters from stored procedures and functions
- Automatic [retrieval of default field values](#)
- [Deferred detail DataSet refresh](#) in master/detail relationships
- [MIDAS](#) technology support
- [MyDataAdapter](#) 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
- Extended data management with [TMyDump](#), [TMyBackup](#) components

#### Script execution:

- Advanced script execution features with [TMyScript](#) component
- Support for executing [individual statements](#) in scripts
- Support for [executing huge scripts stored in files](#) with dynamic loading
- [Optimized multi-statement script execution](#)
- Ability to use standard MySQL client tool syntax in scripts
- Ability to [break long-running query execution](#)

#### SQL execution monitoring:

- Extended SQL tracing capabilities provided by [TMySQLMonitor](#) component and [DBMonitor](#)
- Borland SQL Monitor support
- Ability to [send messages to DBMonitor](#) from any point in your program
- Ability to [retrieve information about the last query execution](#)

#### Visual extensions:

- Includes source code of enhanced TCRDBGrid data-aware grid control
- Customizable [connection dialog](#)
- [Cursor changes](#) during non-blocking execution

#### Design-time enhancements:

- [DataSet Manager tool](#) to control DataSet instances in the project
- Advanced design-time component and property editors
- Integration with [dbForge Fusion for MySQL](#) for browsing database schemas, manipulating database objects and visual building of queries
- Automatic design-time component linking
- Easy [migration from BDE](#) with [Migration Wizard](#)
- More convenient data source setup with the [TMyDataSource](#) component
- Syntax highlighting in design-time editors

#### dbForge Fusion for MySQL main features

- Integration with MyDirect .NET for enhanced component designers and drag-and-drop features
- Stored routines and SQL script debugger
- SQL code completion and navigation
- Visual query builder
- Database Explorer
- Visual object editors
- Database search engine
- Code template library
- Security Manager
- Session Manager
- Export/Import Wizards

**Product clarity:**

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

**Licensing and support:**

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

## 5 What's New

### **05-Sep-12 New Features in MyDAC 7.5:**

- Rad Studio XE3 is supported
- Windows 8 is supported

### **21-Jun-12 New Features in MyDAC 7.2:**

- Update 4 Hotfix 1 for RAD Studio XE2, Delphi XE2, and C++Builder XE2 is now required
- Data Type Mapping support is added
- Data encryption in a client application is added
- The TMyEncryptor component for data encryption is added
- Calling of the TCustomDASQL.BeforeExecute event is added

### **23-Nov-11 New Features in MyDAC 7.1:**

- Update 4 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 0.9.30.4 and FPC 2.6.0 are supported
- Mac OS X in La arus is supported
- Linux x64 in La arus is supported
- FreeBSD in La arus is supported
- Performance of SQL query generation for stored procedure execution is improved

### **15-Sep-11 New Features in Data Access Components for MySQL 7.00:**

- 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

### **28-Apr-11 New Features in Data Access Components for MySQL 6.10:**

- La arus 0.9.30 and FPC 2.4.2 is supported
- Now the BreakExec method also stops working when getting record count if QueryRecCount=True

### **13-Sep-10 New Features in Data Access Components for MySQL 6.00:**

- Embarcadero RAD Studio XE supported

### **10-Sep-09 New Features in Data Access Components for MySQL 5.90:**

- Embarcadero RAD Studio 2010 supported

### **02-Apr-09 New Features in Data Access Components for MySQL 5.80:**

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

### **23-Oct-08 New Features in Data Access Components for MySQL 5.70:**

- 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 La arus
- Completed with more comprehensive structured Help

### **21-Aug-08 New Features in Data Access Components for MySQL 5.55:**

- dbForge Fusion for MySQL support added

### **23-May-08 New Features in Data Access Components for MySQL 5.50:**

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

- Added ability to specify key fields for a dataset
- Added support of automatic records locking

#### 27-Sep-07 New Features in Data Access Components for MySQL 5.20:

- CodeGear RAD Studio 2007 supported
- Added the [OnProgress](#) event in [TMyLoader](#)

#### 12-Jun-07 New Features in MySQL Data Access Components 5.10:

- C++Builder 2007 supported

#### 22-Mar-07 New Features in MySQL Data Access Components 5.00:

##### New functionality:

- Delphi 2007 for Win32 supported
- Implemented [Disconnected Model](#) for working offline and automatically connecting and disconnecting
- Implemented [Local Failover](#) for detecting connection loss and implicitly re-executing some operations
- Support for SSH protocol via [SecureBridge](#) component set added
- Added [DataSet Manager](#) to control project datasets
- Integration with [MyDeveloper Tools 2.00](#) added
- New [TCRBatchMove](#) component for transferring data between all types of TDataSet descendants added
- Output parameters from stored procedures and functions retrieval supported
- Data [export](#) and [import](#) to/from XML supported
- WideMemo field type in Delphi 2006 supported
- [AutoRefresh](#) mode support added
- Option to break long-duration query execution added
- Ability to [search for installed MySQL servers](#) on the network added
- Support for [sending messages](#) to DBMonitor from any point in your program added

##### Support for more MySQL server functionality:

- [HANDLER](#) syntax support in TMyTable added
- [Enumeration value retrieval](#) for ENUM and SET fields added

##### 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
- [TMyScript](#) 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 MySQL client tool syntax added
- Working with [calculated and lookup fields](#) improvements:
- Local [sorting](#) and filtering added
- Record location speed increased
- Improved working with lookup fields
- Greatly increased [performance of applying updates](#) in [CachedUpdates](#) mode
- [Connection pool](#) functionality improvements:
- Efficiency significantly improved
- [API for draining the connection pool](#) added
- Option to [ignore or replace records](#) with duplicated key values in [TMyLoader](#) added
- Enhanced [TMyServerControl](#) functionality for working with [server values](#)
- Ability to customize update commands by attaching external components to [TMyUpdateSQL](#) 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 MySQL Data Access Components 4.40:

- Optimized TDA Loader.CreateColumns method
- Support for Professional editions of Turbo Delphi, Turbo Delphi for .NET, Turbo C++ added
- Added support for macros names in which first symbol is digit
- Added capability to use quoted field names in IndexFieldNames property

**18-May-06 New Features in MySQL Data Access Components 4.30.1:**

- MyDAC is now compatible with InterBase Data Access Components
- Modifying FieldDefs in TVirtualTable component accelerated
- Performance of SaveToFile and LoadFromFile functions in TVirtualTable improved

**26-Jan-06 New Features in MySQL Data Access Components 4.30:**

- Support for Delphi 2006 added
- BIT fields of MySQL 5.0 and above are now represented as TLargeintField
- FastReport 3.20 support added
- Added server version checking on Prepare method
- Added capability to close DataSet quicker when FetchAll property is False
- Improved performance of updating recordsets with multiple fields
- TCustomDADataset.Locate now centers position in DBGrid
- Added support for MIDAS TDataSet.PSExecuteStatement ResultSet parameter

**07-Dec-05 New Features in MySQL Data Access Components 4.00.2:**

- Added 'delimiter' keyword support in TMyScript
- TCustomDADataset.FindKey, TCustomDADataset.FindNearest methods added for BDE compatibility
- Added BIT and INTEGER types support in parameters of stored procedures

**02-Sep-05 New Features in MySQL Data Access Components 4.00.1:**

- Deferred detail dataset refresh feature with TCustomDADataset.Options.DetailDelay property added
- TCustomMyConnection.Ping behavior in case connection loss improved
- FieldDefs.Update behavior in case of temporary tables improved
- Added capability to prevent simultaneous access of several MyEmbConnection instances to single data folder

**29-Jul-05 New Features in MySQL Data Access Components 4.00:**

- Unicode support added
- Enhanced support for Embedded MySQL Server with TMyEmbConnection component added
- Binary protocol support for MySQL 4.1 and above added
- Encrypted SSL connections support with OpenSSL added
- Enhanced support for national charsets added with TMyConnectionOptions.Charset property
- BLOB compression support added
- RefreshQuick for TCustomMyDataSet added
- Retrieve field's default value added with TCustomMyDataSet.Options.DefaultValues property
- Large amount of data support for TMyDump added
- Server start/stop ability for TMyServerControl added
- TMyBuilder component added for easy using SQL Builder for MySQL at run-time
- Optimized macros processing
- FAQ added
- Tested with MySQL server 5.0.9

**30-May-05 New Features in MySQL Data Access Components 3.55:**

- MySQL 5.0.3 BIT type support added
- Optimized MySQLMonitor BLOB parameters processing
- Ability of automatic preparing query with TCustomDADataset.Options.AutoPrepare property added
- Ability to synchronize position at different DataSets with TCustomDADataset.GotoCurrent method added

**21-Jan-05 MySQL Data Access Components 3.50:**

- Support for Delphi 2005 added
- TMySQLMonitor.OnSQL can return statement encoded to an escaped SQL string
- Support for ConnectionTimeout in TMyConnection.ExecSQL added
- CommandTimeout default value set to 0 (infinite)
- TCustomDADataset.UpdateObject support for MIDAS added
- Lock Demo added
- DECIMAL column type in MySQL 5.0.3 support added
- Update Pack 3 is required for Delphi 8

**21-Oct-04 New Features in MySQL Data Access Components 3.30:**

- Full support for all current authentication protocols added
- Generating update SQL for tables from other database added
- TCustomMyDataSet.Options.EnableBoolean property added
- TMyConnection.ThreadId property added



- lxPartialCompare option for DataSet.LocateEx added
- FastReport3 engine and demo added
- Ability to store only a part of data in TMyDump.BackupQuery added
- Creating additional connection for TMyDump disabled
- TCustomMyDataSet.CommandTimeout property added
- "True" value for boolean fields and parameters stored as "1"

**10-Sep-04 New Features in MySQL Data Access Components 3.10.2:**

- Common class DADDataAdapter isolated to Devart.Dac.AdoNet.dll library

**22-Jul-04 New Features in MySQL Data Access Components 3.10.1 new features:**

- Assembly Devart.MyDac.Data renamed to Devart.MyDac.AdoNet
- Fatal errors processing improved
- TINYINT(1) fields now represented as TBooleanField

**08-Jul-04 New Features in MySQL Data Access Components 3.10:**

- Local sorting ability with TMemDataSet.IndexFieldNames added
- TCustomMyTable.IndexDefs property added
- TMyConnection.Options.NumericType property added
- TMyStoredProc component added
- MyDataAdapter component added

**29-Apr-04 New Features in MySQL Data Access Components 3.00.1:**

- TCustomMyDataSet.Options.LongStrings property added
- TMyLoader.OnPutData event published again
- Trial version IDE warning disabled
- TCRColumn.TotalValue property added

**09-Apr-04 New Features in MySQL Data Access Components 3.00:**

- Support for Delphi 8 added
- Connection pooling support
- Performance improved
- TMyLoader performance greatly improved
- TCRGrid sources in Standard edition
- .NET Windows Forms demo project added
- ASP.NET demo project added
- Global variable MySQLClientLibrary added
- New time trial limitation

**05-Feb-04 New Features in MySQL Data Access Components 2.00.3:**

- SELECT " support added
- Method TMyConnection.Ping added
- Method TMyConnection.GetExecuteInfo added
- Mouse wheel support added to CRDBGrid
- Embedded MySQL Server Demo added
- ConnectDialog Demo added

**30-Dec-03 New Features in MySQL Data Access Components 2.00.2:**

- BDE Migration Wizard algorithm optimized
- Limited MySQL server 4.1.1 support added
- If libmysql.dll not found then raise EOSError (instead of Exception)
- Property TCustomMyDataSet.InsertId: int64 added
- timestamp support added for CheckRowVersion = True

**24-Nov-03 New Features in MySQL Data Access Components 2.00.1:**

- Property MyConnection.Options.Direct is set to True by default
- TCustomMyDataSet.Lock method added
- AutoInc fields can be modified now

**02-Oct-03 New Features in MySQL Data Access Components 2.00:**

- Access to MySQL without client library using DirectMySQLObjects by Cristian Nicola
- Prepare support and new parameter binding schema for MySQL 4.1 added
- Supports working with MySQL server and Embedded server at the same time
- BDE migration wizard
- TMyDump component to store a database or its parts as a script
- TMyBackup component for backup copying specified tables on the server

- TMyServerControl component to manage the server and standard service tasks execution
- TMyLoader component for fast loading data to the server
- New options of TMyConnection such as Compress, Protocol, Direct and Embedded added
- New properties ClientVersion, ServerVersion were added to TMyConnection
- Method ExecSQL in TMyConnection added
- Methods GetTableNames and GetDatabaseNames in TMyConnection added
- Property TMyConnection.Charset added
- Property TMyConnection.IsolationLevel added
- Methods LockTable and UnlockTable added to TCustomMyDataset
- Properties Limit and Offset added to TCustomMyTable
- Method TCustomMyTable.EmptyTable added
- FetchAll set to True by default
- Large SQL (INSERT/UPDATE BLOB's) executing performance greatly improved

**06-Jun-03 New Features in MySQL Data Access Components 1.50:**

- Embedded MySQL Server support added
- MySQL Server 4.1 limited support added
- Properties Port and Database in ConnectForm added
- RefreshRecord performance improved
- InfoPower demos added
- 'Explain query...' added to design-time MyQuery menu
- 'Show CREATE...' added to design-time MyQuery and MyTable menus
- SQL Generator improved - support for complicated statements added
- SQL Generator improved - "Quote names" checkbox added
- Complex keys support added
- Design-time SQL Generator was simplified
- TParam -> TDAParam
- Embedded MySQL Server support added for Kylix
- Check for datadir present added
- Changed behavior on calculating affected rows count

**04-Apr-03 New Features in MySQL Data Access Components 1.30.2:**

- Unit MySQLAccess renamed to MyClasses
- Property TMyDataSetOptions.LongStrings removed
- Parameters parsing improved. Symbol ':' in string literals is ignored
- Search algorithm for 'libmysqlclient.so' under Linux improved

**24-Feb-03 New Features in MySQL Data Access Components 1.30.1:**

- Refresh improved - current record is restored after Refresh call
- Property MyConnection.Options.KeepDesignConnected added
- Property MyConnectDialog.StoreLogInfo published
- Property MyScript.DataSet was published
- Property TMyCommand.InsertId: int64 added
- TINYTEXT -> TMemoField, TINYBLOB -> TBlobField
- Support for TIMESTAMP (10), TIMESTAMP (4), TIMESTAMP (2) added
- Support for LIKE expressions in Filter property added (D2706)

**30-Jan-03 New Features in MySQL Data Access Components 1.30:**

- MySQL v4.0 support added
- Dataset 'with many fields' update performance improved
- Improved performance for opening queries with lot of parameters

**26-Dec-02 New Features in MySQL Data Access Components 1.20:**

- Kylix2 and Kylix3 support
- ReportBuilder demos added
- DBMonitor client implementation moved to COM server
- Fetch performance improved for DataSet.FetchAll = True
- 'Connection Lost' error processing improved

**08-Oct-02 New Features in MySQL Data Access Components 1.10:**

- Delphi 7 support
  - New memory management model for ftString and ftVarBytes types. Allows significantly decrease memory usage on large tables fetch. Controlled by FlatBuffers dataset option
  - Support for blob fields in CachedUpdates mode
-



## 6 Demo Projects

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

The MyDAC demo projects consist of one large project called *MyDacDemo* with demos for all main MyDAC components, use cases, and data access technologies, and a number of smaller projects on how to use MyDAC in different IDEs and how to integrate MyDAC with third-party components. Most demo projects are built for Delphi, Borland Developer Studio, and Kylix. There are only two MyDAC demos for C++Builder. However, the C++Builder distribution includes source code for all the other demo projects as well.

### Where are the MyDAC demo projects located?

In most cases all the MyDAC demo projects are located in "%MyDac%\Demos\".

In Delphi 2007 for Win32 under Windows Vista all the MyDAC demo projects are located in "My Documents\Devart\MyDac for Delphi 2007\Demos", for example "C:\Documents and Settings\All Users\Documents\Devart\MyDac 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
|   | -MyDacDemo [.NET version of the main MyDAC demo project]
|   |   | -TechnologySpecific
|   |   |   | - Embedded [.NET version of the Embedded MySQL server using demo]
|   |   |   | - SecureBridge [.NET version of A component and a demo for integration with the
SecureBridge library]
|   |   | -Miscellaneous
|   |   | - [Some other .NET demo projects]
| -Win32
|   | -MyDacDemo [Win32 version of the main MyDAC demo project]
|   |   | -ThirdParty
|   |   |   | - [A collection of demo projects on integration with third-party components]
|   |   |   | -TechnologySpecific
|   |   |   |   | - Embedded [Win32 version of the Embedded MySQL server using demo]
|   |   |   |   | - SecureBridge [A component and a demo for integration with the SecureBridge
library]
|   |   |   | -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 is omitted. For these IDEs you will see the following structure.

```
Demos
| -MyDacDemo [The main MyDAC demo project]
| -TechnologySpecific
|   | - Embedded [Win32 version of the Embedded MySQL server using demo]
|   | - SecureBridge [A component and a demo for integration with the SecureBridge
library]
| -ThirdParty
|   | - [A collection of demo projects on integration with third-party components]
| -Miscellaneous
|   | - [Some other demo projects on design technologies]
```

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

**Note:** This documentation describes ALL the MyDAC demo projects. The actual demo projects you will have installed on your computer depends on your MyDAC version, MyDAC 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 MyDAC demo projects

To explore a MyDAC demo project,

1. Launch your IDE.

2. In your IDE, choose File → Open Project from the menu bar.
3. Find the directory you installed MyDAC 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 included sample applications are fully functional. To use the demos, you have to first set up a connection to MySQL. 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 database.

**Note:** The MyDAC demo directory includes two sample SQL scripts for creating and dropping all the test schema objects used in the MyDAC 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 MyDAC demo project in the [Getting Started](#) topic. The other MyDAC demo projects include a *ReadMe.txt* file with individual building and launching instructions.

## Demo project descriptions

### MyDacDemo

MyDacDemo 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 MyDAC components.

#### **General demos**

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

#### **MySQL-specific demos**

A collection of samples that demonstrate how to incorporate MySQL features in database applications.

*MyDacDemo* can be opened from %MyDac%\Demos\MyDacDemo\MyDacDemo.dpr (.bdsproj). The following table describes all demos contained in this project.

### Working with Components

| Name                 | Description  |
|----------------------|--|
| <b>Command</b>       | Uses <a href="#">TMyCommand</a> to execute SQL statements. Demonstrates how to execute commands in a separate thread, and how to <a href="#">break long-duration query execution</a> .   |
| <b>ConnectDialog</b> | Demonstrates how to customize the <a href="#">MyDAC connect dialog</a> . Changes the standard MyDAC 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 MyDAC 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>Loader</b>        | Uses the <a href="#">TMyLoader</a> component to quickly load data into a server table. TMyLoader loads data by grouping several data rows into a single INSERT statement and executing this statement. This way is much faster than executing one INSERT statement per row. This demo also compares the two TMyLoader data loading handlers: <a href="#">GetColumnData</a> and <a href="#">PutData</a> . |
| <b>Query</b>         | Demonstrates working with <a href="#">TMyQuery</a> , which is one of the most useful MyDAC components. Includes many TMyQuery usage scenarios. Demonstrates how to edit data and export it to XML files.<br>Note: This is a very good introductory demo. We recommend starting here when first becoming familiar with MyDAC.   |
| <b>StoredProc</b>    | Uses <a href="#">TMyStoredProc</a> to access an editable recordset from an MySQL stored procedure in the client application.   |
| <b>Table</b>         | Demonstrates how to use <a href="#">TMyTable</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="#">TMyUpdateSQL</a> component to customize update commands. Lets you optionally use <a href="#">TMyCommand</a> and <a href="#">TMyQuery</a> objects for carrying out insert, delete, query, and update commands.   |

**VirtualTable** Demonstrates working with the [TVirtualTable](#) 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 MyDAC'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 MyDAC 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>Pictures</b>       | Uses MyDAC functionality to work with BLOB fields and graphics. The sample demonstrates how to retrieve binary data from MySQL database and display it on visual components. The sample also shows how to load and save pictures to files and to the database.                                   |
| <b>Text</b>           | Uses MyDAC functionality to work with text. The sample demonstrates how to retrieve text data from MySQL database and display it on visual components. The sample also shows how to load and save text to files and to the database.   |
| <b>Transactions</b>   | Demonstrates the recommended approach for managing transactions with the <a href="#">TMyConnection</a> component. The TMyConnection interface provides a wrapper for MySQL server commands like START TRANSACTION, COMMIT, ROLLBACK.   |

#### MySQL-specific Demos

| Name        | Description   |
|-------------|---|
| <b>Lock</b> | Demonstrates two kinds of <a href="#">row-level locking</a> (immediate locking and delayed locking) with the InnoDB storage engine. This functionality is based on the following MySQL commands: SELECT ... FOR UPDATE and SELECT ... LOCK IN SHARE MODE. |

#### Supplementary Demo Projects

MyDAC also includes a number of additional demo projects that describe some special use cases, show how to use MyDAC in different IDEs and give examples of how to integrate it with third-party components. These supplementary MyDAC demo projects are sorted into subfolders in the %MyDac%\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="#">MyDataAdapter</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 MyDAC to create a WinForms application. This demo project creates a simple WinForms application and fills a data grid from an <a href="#">MyDataAdapter</a> data source.   |
| MyDacDemo           | <b>MyDacDemo</b> | [.NET version of the main MyDAC demo project - see above]   |
| Technology Specific | <b>Embedded</b>  | Demonstrates <a href="#">working with Embedded MySQL server</a> by using the <a href="#">TMyEmbConnection</a> component. This demo creates a database structure, if it does not already exist, opens a table from this database. Also this demo shows how to process the log messages of the Embedded server. |
| <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 MyDAC can be used with FastReport components. This project consists of two parts. The first part is several packages that integrate MyDAC components into the FastReport editor. The second part is a demo application that lets you design and preview reports with MyDAC technology in the FastReport editor.  |
|                     | <b>InfoPower</b>     | Uses InfoPower components to display recordsets retrieved with MyDAC. This demo project displays an InfoPower grid component and fills it with the result of an MyDAC query. Shows how to link MyDAC data sources to InfoPower components.  |
| ThirdParty          | <b>IntraWeb</b>      | A collection of sample projects that show how to use MyDAC 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>QuickReport</b>   | Lets you launch and view a QuickReport application based on MyDAC. This demo project lets you modify the application in design-time.  |
|                     | <b>ReportBuilder</b> | Uses MyDAC data sources to create a ReportBuilder report that takes data from MySQL database. Shows how to set up a ReportBuilder document in design-time and how to integrate MyDAC components into the Report Builder editor to perform document design in run-time.  |
|                     | <b>Embedded</b>      | Demonstrates <a href="#">working with Embedded MySQL server</a> by using the <a href="#">TMyEmbConnection</a> component. This demo creates a database structure, if it does not already exist, opens a table from this database. Also this demo shows how to process the log messages of the Embedded server.   |
| Technology Specific | <b>SecureBridge</b>  | <p>The demo project demonstrates how to integrate the <a href="#">SecureBridge</a> components with MyDAC to ensure secure connection to MySQL server through an SSH tunnel and SSL.</p> <p>This demo consists of three parts. The first part is a package that contains TMySSHIOHandler and TMySSLIOHandler component. These components provide integration with the <a href="#">SecureBridge</a> library. The second part is two sample projects that demonstrate how to connect to MySQL server through an SSH server and through SSL, connect to the SSH server with SecureBridge by password or by public key, generate reliable random numbers, enable local port forwarding.</p> <p>For more information see the <i>Readme.html</i> file in the demo directory.</p> |













|                  |                       |   |
|------------------|-----------------------|---|
|                  | <b>CBuilder</b>       | A general demo project about how to create MyDAC-based applications with C++Builder. Lets you execute SQL scripts and work with result sets in a grid. This is one of the two MyDAC demos for C++Builder.   |
|                  | <b>Dll</b>            | Demonstrates creating and loading DLLs for MyDAC-based projects. This demo project consists of two parts - an My Dll project that creates a DLL of a form that sends a query to the server and displays its results, and an My Exe project that can be executed to display a form for loading and running this DLL. Allows you to build a dll for one MyDAC-based project and load and test it from a separate application. |
| Miscellaneous    | <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.              |
|                  | <b>Midas</b>          | Demonstrates using MIDAS technology with MyDAC. 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  |
| <b>MyDacDemo</b> | <b>MyDacDemo</b>      | [ in32 version of the main MyDAC demo project - see above   |









## 7 Component List

This topic presents a brief description of the components included in the Data Access Components for MySQL library. Click on the name of each component for more information. These components are added to the MySQL 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 MyDAC components are included in all MyDAC editions. MyDAC Professional and Developer Edition components are not included in MyDAC Standard Edition.

### Basic MyDAC components

|   |                                  |  |
|---|----------------------------------|--|
|    | <a href="#">TMyConnection</a>    | Lets you set up and control connections to MySQL database server.  |
|    | <a href="#">TMyQuery</a>         | Uses SQL statements to retrieve data from MySQL table or tables and supply it to one or more data-aware components through a TDataSource component. Provides flexible data update functionality. |
|    | <a href="#">TMyCommand</a>       | Executes SQL statements and stored procedures, which do not return rowsets.  |
|    | <a href="#">TMyTable</a>         | Lets you retrieve and update data in a single table without writing SQL statements.  |
|    | <a href="#">TMyStoredProc</a>    | Executes stored procedures and functions.  |
|   | <a href="#">TMyUpdateSQL</a>     | Lets you tune update operations for a DataSet component.   |
|  | <a href="#">TMyDataSource</a>    | Provides an interface between MyDAC dataset components and data-aware controls on a form.  |
|  | <a href="#">TMyScript</a>        | Executes sequences of SQL statements.  |
|  | <a href="#">TMySQLMonitor</a>    | Interface for monitoring dynamic SQL execution in MyDAC-based applications.  |
|  | <a href="#">TMyConnectDialog</a> | Used to build custom prompts for username, password and server name.   |
|  | <a href="#">TVirtualTable</a>    | Dataset that stores data in memory. This component is placed on the Data Access page of the Component palette, not on the MySQL Access page.   |
|  | <a href="#">MyDataAdapter</a>    | .NET component, uses TDataSet as data source for retrieving and saving data to System.Data.DataSet.  |

### MyDAC Professional and Developer Edition components

|   |                                  |  |
|---|----------------------------------|--|
|  | <a href="#">TMyEncryptor</a>     | Represents data encryption and decryption in client application.                                       |
|  | <a href="#">TMyLoader</a>        | Provides quick loading data to MySQL database.   |
|  | <a href="#">TMyDump</a>          | Serves to store a database or its parts as a script and also to restore database from received script. |
|  | <a href="#">TMyBackup</a>        | Serves for backup copying specified tables on the server.  |
|  | <a href="#">TMyServerControl</a> | Serves to control the server and execution of standard service tasks.                                  |
|  | <a href="#">TMyEmbConnection</a> | Is used to establish connection to Embedded MySQL server.  |

[TMyBuilder](#)

Serves to manage SQL Builder for MySQL Add-in.

[TMyMetaData](#)

Retrieves metadata on specified SQL object.

[TCRBatchMove](#)

Transfers data between all types of TDataSet descendants. This component is placed on the Data Access page of the Component palette, not on the MySQL Access page.

**See Also**

- [Hierarchy chart](#)
-

## 8 Hierarchy Chart

Many MyDAC classes are inherited from standard VCL/CLX classes. The inheritance hierarchy chart for MyDAC is shown below. The MyDAC 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
|   |   |   |-TCustomMyConnection
|   |   |   |   |-TMyConnection
|   |   |   |   |-TMyEmbConnection
|   |   |-TDataSet
|   |   |   |-TMemDataSet
|   |   |   |   |-TCustomDADataSet
|   |   |   |   |   |-TCustomMyDataSet
|   |   |   |   |   |   |-TMyQuery
|   |   |   |   |   |   |-TCustomMyTable
|   |   |   |   |   |   |   |-TMyTable
|   |   |   |   |   |   |   |-TCustomMyStoredProc
|   |   |   |   |   |   |   |   |-TMyStoredProc
|   |   |   |   |   |   |   |   |-TMyServerControl
|   |   |   |   |   |-TDAMetaData
|   |   |   |   |   |   |-TMyMetaData
|   |   |   |   |-TVirtualTable
|   |   |-TDataSource
|   |   |   |-TCRDataSource
|   |   |   |-TMyDataSource
|   |-DADDataAdapter
|   |   |-MyDataAdapter
|   |-TCRBatchMove
|   |-TCustomConnectDialog
|   |   |-TMyConnectDialog
|   |-TCustomDASQL
|   |   |-TMyCommand
|   |-TCustomDASQLMonitor
|   |   |-TMySQLMonitor
|   |-TDADump
|   |   |-TMyDump
|   |-TDALoader
|   |   |-TMyLoader
|   |-TDAScript
|   |   |-TMyScript
|   |-TMyBackup
|   |-TMyBuilder
|   |-TMyIOHandler
|   |-TCREncryptor
|   |   |-TMyEncryptor

```

## 9 Requirements

### Requirements for using MyDAC in Direct mode

If you use MyDAC to connect to MySQL in [Direct](#) mode, you do not need to have MySQL client library on your machine or deploy it with your MyDAC-based application.

### Requirements for using MyDAC in Client mode

If you use MyDAC to connect to MySQL in Client mode, you need to have access to the MySQL client library. In particular, you will need to make sure that the MySQL client library is installed on the machines your MyDAC-based application is deployed to. MySQL client library is libmysql.dll file for Windows, or libmysqlclient.so (libmysqlclient.so.X) for Linux. Please refer to descriptions of LoadLibrary() and dlopen() functions accordingly for detailed information about MySQL client library file location. You may need to deploy the MySQL client library with your application or require that users have it installed.

### Requirements for using Embedded MySQL server

If you are working with Embedded server, you should have access to Embedded MySQL server library (libmysqld.dll). For more information visit [Using Embedded server](#).

---

# 10 Compatibility

## MySQL Compatibility

MyDAC supports the following database servers:

- MySQL servers: 6.0, 5.5, 5.1, 5.0, 4.1, 4.0, and 3.23
- MySQL Embedded servers: 6.0, 5.5, 5.1, 4.1, and 4.0

## IDE Compatibility

MyDAC is compatible with the following IDEs:

- Embarcadero RAD Studio XE3
  - 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 OSX32
- Embarcadero RAD Studio XE2 (Requires [Update 4 Hotfix 1](#))
  - [Embarcadero Delphi XE2 for Win32, Win64, and OSX32](#)
  - Embarcadero Delphi XE2 for Win64
  - Embarcadero Delphi XE2 for OSX32
  - [Embarcadero C++Builder XE2 for Win32 and OSX32](#)
  - 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](#) 0.9.30.4 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 MyDAC additionally supports Starter Edition.

La arus and Free Pascal are supported only in Trial Edition and in Professional and Developer editions with source code.

## Supported Target Platforms

- Windows, 32-bit and 64-bit
- Mac OS X
- iOS (only in Delphi XE2 in Professional and Developer edition with source code)

- 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 was introduced in Delphi XE2, and is not available in C++Builder and older versions of Delphi. Support for Mac OS X was introduced in Delphi XE2 and C++Builder XE2, and is not available in older versions of Delphi and C++Builder.

### **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.

### **dbForge Fusion for MySQL Compatibility**

The current version of MyDAC is compatible with dbForge Fusion 4.xx for RAD Studio 2007 - XE2

---

# 11 Installation

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

Compiled versions of MyDAC are installed automatically by the MyDAC Installer for all supported IDEs except for Kylix and Lazarus. Versions of MyDAC with Source Code must be installed manually.

Installation of MyDAC from sources is described in the supplied *ReadMeSrc.txt* file.

## Before installing MyDAC ...

Two versions of MyDAC 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 MyDAC may be incompatible with older versions of ODAC, IBDAC, and SDAC.

So before installing a new version of MyDAC, uninstall any previous version of MyDAC 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 [mydac@devart.com](mailto:mydac@devart.com)

**Note:** You can avoid performing MyDAC uninstallation manually when upgrading to a new version by directing the MyDAC 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 `mydacXX.exe /force`, specifying the full path to the appropriate version of the installation program).

## Installed packages

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

**Note:** %MyDAC% denotes the path to your MyDAC 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       |
| mydacXX.bpl       | MyDAC run-time package    | Windows\System32 |
| dclmydacXX.bpl    | MyDAC design-time package | Delphi\Bin       |
| dclmysqlmonXX.bpl | TMySQLMonitor component   | Delphi\Bin       |
| mydacvclXX.bpl*   | VCL support package       | Delphi\Bin       |
| crcontrolsXX.bpl  | TCRDBGrid component       | Delphi\Bin       |

\* Not included in 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                | %MyDAC%\Bin           |
| Devart.Dac.AdoNet.dll   | Data provider core package             | Delphi\Bin            |
| Devart.MyDac.dll        | MyDAC Delphi for .NET run-time package | Global Assembly Cache |
| Devart.MyDac.Design.dll | MyDAC design-time package              | %MyDAC%\Bin           |
| Devart.Vcl.dll          | TCRDBGrid component                    | Global Assembly Cache |
| Devart.MyDac.AdoNet.dll | Data provider for MySQL package        | Global Assembly Cache |

### Additional packages for using MyDAC managers and wizards

| <i>Name</i> | <i>Description</i> | <i>Location</i> |
|-------------|--------------------|-----------------|
|-------------|--------------------|-----------------|

|                      |                            |             |
|----------------------|----------------------------|-------------|
| datasetmanagerXX.bpl | DataSet Manager package    | Delphi\Bin  |
| mymigwi_ardXX.dll    | MyDAC BDE Migration wi_ard | %MyDAC%\Bin |

#### Additional .NET packages for using MyDAC managers and wi\_ards

| <i>Name</i>                 | <i>Description</i>                  | <i>Location</i>       |
|-----------------------------|-------------------------------------|-----------------------|
| Devart.Dac.DsManager.dll    | DataSet Manager Assembly            | Global Assembly Cache |
| Devart.MyDac.MigWi_ard.dll* | MyDAC BDE Migration wi_ard Assembly | Global Assembly Cache |

\* Included in Borland Delphi 8 only

### **Environment Changes**

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

#### Delphi

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

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

#### Delphi for .NET

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

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

#### C++Builder

##### **C++Builder 5, 6:**

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

##### **C++Builder 2006, 2007:**

- \$(BCB)\MyDAC\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)\MyDAC\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 MyDAC Installer performs C++Builder environment changes automatically for compiled versions of MyDAC.

#### Kylix

Kylix the only IDE which you will have to configure manually to use both compiled MyDAC libraries and versions of MyDAC with Source Code. Complete the following steps to configure your Kylix environment. Replace %MyDAC% with the path to your MyDAC installation directory.

1. Make MyDAC packages reachable for Kylix. Add the directory where MyDAC packages are installed to LD\_LIBRARY\_PATH  

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:%MyDAC%
```

Alternatively, you can copy all the MyDAC packages (\*.so) to any directory reachable by Kylix, such as kylix/bin.
2. Install MyDAC in Kylix. Select Component Install Packages from the Kylix menu. Press Add button and select bpldclmydacX.so.X.XX package. On pressing OK, the MyDAC components will be available in the MySQL Access group.
3. Add %MyDAC%/lib directory to the Search Path of your project.

#### La arus

The MyDAC installation program only copies MyDAC files. You need to install MyDAC packages to La arus



IDE manually. Open %MyDAC%\Source\Lazarus1\dclmydac.lpk file in Lazarus and press the Install button. After that Lazarus IDE will be rebuilt with MyDAC packages.

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

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

### **Installation of Additional Components and Add-ins**

#### dbForge Fusion for MySQL

dbForge Fusion for MySQL is a powerful database development and administration tool for MySQL.

dbForge Fusion for MySQL is available as an add-in for Delphi and C++Builder 2009, CodeGear RAD Studio 2007, or as a standalone application. For more information, visit the [dbForge Fusion for MySQL page online](#).

#### MyBuilder

MyBuilder is an easy to use and versatile MyDAC design-time extension to manipulate data and database objects of MySQL. With MyBuilder Add-in you can build, execute, verify and optimize your SQL statements. For more information, visit the [MyBuilder page online](#).

#### DBMonitor

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

## 12 Deployment

MyDAC 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 MyDAC-based applications built without run-time packages, provided you are using a registered version of MyDAC.

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 MyDAC 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 |
| mydacXX.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  |
| mydacXX.bpl      | always  |
| dacvclXX.bpl     | if your application uses the MyDacVcl unit      |
| mydacvclXX.bpl   | if your application uses the MyDacVcl unit      |
| crcontrolsXX.bpl | if your application uses the CRDBGrid component |

### Deploying .NET applications

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

|                             |  |
|-----------------------------|--|
| Devart.Dac.dll              | always   |
| Devart.MyDac.dll            | always   |
| Devart.Dac.<br>AdoNet.dll   | If your application uses MyDataAdapter component |
| Devart.MyDac.<br>AdoNet.dll | If your application uses MyDataAdapter 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.

## 13 Licensing and Subscriptions

Data Access Components for MySQL 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 MyDAC installation folder.

### Licensing

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

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

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 MyDAC Trial Edition under a temporary **Evaluation License**, which allows you to test MyDAC Trial Edition for a period of 60 days, after which you must either remove all files associated with MyDAC or purchase a full license.

Licenses can be purchased for the following editions of MyDAC: MyDAC Standard Edition, MyDAC Professional Edition, and MyDAC Professional Edition with Source Code, MyDAC Developer Edition, and MyDAC Developer Edition with Source Code. An edition comparison chart can be found [here](#).

To purchase a license for MyDAC, please visit [www.devart.com/mydac/ordering.html](http://www.devart.com/mydac/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 MyDAC: MyDAC Standard Edition, MyDAC Professional Edition, and MyDAC Professional Edition with Source Code, MyDAC Developer Edition, and MyDAC Developer Edition with Source Code.

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

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

### Subscriptions

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

Users with a valid MyDAC Subscription get the following benefits:

- Product support through the MyDAC [Priority Support](#) program
- Access to new versions of MyDAC when they are released
- Access to all MyDAC 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

MyDAC Evaluation License lets you try MyDAC Trial Edition for a period of 60 days.

There are no functionality limitations in MyDAC Trial Edition during the trial period for most supported IDEs, except the following:

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

## 14 Getting Support

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

### Support Options

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

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

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

### MyDAC Priority Support

MyDAC Priority Support is an advanced product support service for getting expedited individual assistance with MyDAC-related questions from the MyDAC 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 [MyDAC Subscription](#).

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

- The version of Delphi, C++Builder or Kylix you are using.
  - Your MyDAC Registration number.
  - Full MyDAC edition name and version number. You can find both of these in the About sheet of TMyConnection Editor or from the MySQL About menu.
  - Versions of the MySQL server and client you are using.
  - A detailed problem description.
  - If possible, a small test project that reproduces the problem. Please include definitions for all database objects and avoid using third-party components.
-

## 15 Frequently Asked Questions

This page contains a list of Frequently Asked Questions for Data Access Components for MySQL. If you have encounter a question with using MyDAC, please browse through this list first. If this page does not answer your question, refer to the Getting Support topic in MyDAC help.

### Installation and Deployment

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

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

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

You can have such problems if you installed incompatible MyDAC, SDAC, ODAC or IBDAC 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 MyDAC-based applications to work?

Usually, you do not need any additional files. The only exceptions to this rule are listed below:

- o If you are using MySQL Embedded server (if you are using TMyConnection with TMyConnection.Options.Embedded = True or TMyEmbConnection), you need the server itself (*libmysqld.dll*) and the service files for it, for example *errmsg.sys*.
- o If you are connecting in Client mode, (TMyConnection.Options.Direct = False), you need *libmysql.dll*.
- o If you are using SSL (TMyConnection.Options.Protocol = mpSSL), you need the OpenSSL library files - *ssleay32.dll* and *libeay32.dll*.

#### 3. When I try to install MyDAC packages under Kylix, I get an "Invalid package" error.

Probably you are using Kylix Open Edition. MyDAC does not support this version of Kylix.

#### 4. When I try to connect to the server, I get an error "MySQL client library couldn't be loaded."

You are using TMyConnection.Options.Direct := False mode and the client library is not available for your application.

Windows: You should copy client file *libmysql.dll* to a folder available to the executable unit of your program. For example, to the folder containing the executable or to the Windows system folder. For more details, see the description of LoadLibrary and the PATH environment variable.

Linux: You should copy the client file *libmysqlclient.so.X* to the folder available to the executable unit of your program. For more details, see the description of the dlopen function and the LD\_LIBRARY\_PATH environment variable.

#### 5. Core Lab renaming issue that concerns Delphi for .Net users

- o 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 MyDAC?

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

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

You can create your own components that are inherited from MyDAC or that use the MyDAC 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 MyDAC 5.00?

The basic MyDAC license agreement will remain the same. With MyDAC 5.00, the [MyDAC Edition](#)

[Matrix](#) will be reorganized and a new [MyDAC Subscription Program](#) will be introduced.

#### 4. What do the MyDAC 5.00 Edition Levels correspond to?

MyDAC 5.00 will come in six editions: Trial, Standard, Professional, Professional with Sources, Developer, and Developer 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 MyDAC 5.00**

| Old Edition Level          | New Edition Level                       |
|----------------------------|---|
| - No Correspondence        | - MyDAC Standard Edition                |
| MyDAC Standard Edition     | MyDAC Professional Edition              |
| MyDAC Professional Edition | MyDAC Professional Edition with Sources |
| - No Correspondence        | - MyDAC Developer Edition               |
| - No Correspondence        | - MyDAC Developer Edition with Sources  |
| MyDAC Trial Edition        | MyDAC Trial Edition                     |

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

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

After MyDAC 5.00, all upgrades to future versions are free to users with an active MyDAC Subscription.

Users that have a registration for versions of MyDAC prior to MyDAC 5.00 will have to first upgrade to MyDAC 5.00 to jump in on the Subscription program.

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

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

Users with a valid MyDAC Subscription get the following benefits:

- Access to new versions of MyDAC when they are released
- Access to all MyDAC updates and bug fixes
- Product support through the MyDAC Priority Support program
- Notification of new product versions

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

The MyDAC Subscription Program is available for registered users of MyDAC 5.00 and higher.

#### 7. Can I use my version of MyDAC after my Subscription expires?

Yes, you can. MyDAC version licenses are perpetual.

#### 8. I want a MyDAC Subscription! How can I get one?

An annual MyDAC Subscription is included when ordering or upgrading to any registered (non-Trial) edition of MyDAC 5.00 or higher.

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

#### 9. Does this mean that if I upgrade to MyDAC 5 from MyDAC 4, I'll get an annual MyDAC Subscription for free?

Yes.

#### 10. How do I upgrade to MyDAC 5.00?

To upgrade to MyDAC 5.00, you can get a Version Update from the [MyDAC Ordering Page](#). For more information, please contact [sales@crlab.com](mailto:sales@crlab.com).

### Performance

#### 1. How productive is MyDAC?

MyDAC uses a low-level protocol to access the database server. This allows MyDAC to achieve high performance. From time to time we compare MyDAC with other products, and MyDAC always takes first place.

#### 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, MyDAC 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 MyDAC component editors at design time?

Download and install [MySQL Developer Tools](#). In addition to syntax highlighting, MySQL Developer Tools provides a lot of [additional features](#).

Alternatively, you can download and install the freeware [SynEdit component set](#).

### 2. How can I quickly convert a project from BDE to MyDAC?

To quickly migrate your project from BDE you can use the BDE Migration Wizard. To start it, open your project and choose BDE Migration Wizard from the MySQL menu of your IDE.

### 3. How can I determine which version of MyDAC I am using?

You can determine your MyDAC version number in several ways:

- During installation of MyDAC, consult the MyDAC Installer screen.
- After installation, see the *history.html* file in your MyDAC installation directory.
- At design-time, select MySQL About MyDAC from the main menu of your IDE.
- At run-time, check the value of the MydacVersion and DACVersion constants.

### 4. 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.

### 5. How can I execute a query saved in the SQLInsert, SQLUpdate, SQLDelete, or SQLRefresh properties of a MyDAC 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 MyDAC 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.

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

The value of the SQLInsert property will then be used by MyDAC 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 MyDAC dataset object will generate the query itself using the appropriate insert, update, delete, or refresh record syntax.

### 6. How can I get a list of the databases on the server?

Use the TMyConnection.GetDatabaseNames method.

### 7. How can I get a list of the tables list in a database?

Use the TMyConnection.GetTableNames method.

### 8. Some questions about the visual part of MyDAC

The following situations usually arise from the same problem:

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

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

## General Questions

### 1. I would like to develop an application that works with MySQL Server. Should I use MyDAC or DbxMda?

[DbxMda](#) is our dbExpress driver for MySQL. 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 specialised cases dbExpress providers lose some functionality. For example, the dbExpress design-time is quite poor and cannot be expanded.

MyDAC is a specialised set of components for MySQL, which has advanced server-specific design-time and a component interface similar to that of BDE.

We tried to include maximal support of MySQL-specific features in both DbxMda and MyDAC.

However, the nature of dbExpress technology has some insurmountable restrictions. For example, Unicode fields cannot be passed from a driver to dbExpress.

MyDAC and DbxMda use the same kernel and thus have similar performance. 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 MySQL, it is probably better to use DbxMda. In other cases, especially when migrating from BDE or ADO, you should use MyDAC.

## **2. Are the MyDAC connection components thread-safe?**

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

## **3. Behaviour of my application has changed when I upgraded MyDAC. How can I restore the old behaviour with the new version?**

We always try to keep MyDAC compatible with previous versions, but sometimes we have to change behaviour of MyDAC 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 MyDAC help. This topic describes such changes, and how to revert to the old MyDAC behaviour.

## **4. 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 TCustomMyDataSet.Options.StrictUpdate topic in MyDAC help for more information.

## **5. I have problems using BIGINT and INT UNSIGNED fields as key fields in master/detail relationships, and accessing values of such fields through the Field.Value property.**

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 (for more information see the SetVarValue protected method of TLargeIntField in the DB unit). 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.

For master/detail relationships the problem can be avoided only by changing type of the key field to INT, as Delphi's master/detail mechanism works through Field.Value.

## **6. On accessing server I get a 'MySQL server has gone away' or 'Lost connection to MySQL server during query' error.**

First of all, you should find out what causes the problem. The list of most frequent reasons for this error to occur is below.

- o Client side: The value of TMyConnection.ConnectionTimeout or TCustomMyDataSet.CommandTimeout is too small. To check this hypothesis, try setting TCustomMyDataSet.CommandTimeout to 0 (infinite) and TMyConnection.ConnectionTimeout to 300.
- o Server side: MySQL server has closed the connection. You can read a detailed description of all possible reasons for this to happen in the [MySQL Reference Manual](#). Almost always it is because the value of wait\_timeout variable is too small. Try increasing it. If this solution is not possible (for example, because you don't have enough rights), you should invoke MyConnection.Ping with an interval less than wait\_timeout. Use TTimer in TMyConnection thread to accomplish this task.
- o Unstable connection (GPRS etc). In case of unstable connection you can adapt MyDAC to work in such conditions by changing some of its settings. For more information please see the "Working in Unstable Networks" article in the MyDAC help documentation.

If the connection is lost, MyDAC tries to reconnect to server. However, your last command will probably not be executed, and you should repeat it again. MyDAC does not try to reconnect if a transaction has started or if at least one of statements is prepared.

## **7. Some problems using TCustomDADataset.FetchAll=False mode**

The following problems may appear when using FetchAll=False mode:



- I have problems working with temporary tables.
- I have problems working with transactions.
- Sometimes my application hangs on applying changes to the database.

Usage of FetchAll=False mode has many advantages; however, it also has some restrictions since it requires an additional connection to server for data fetching to be created. The additional connection is created to prevent the main connection from blocking.

These problems can be avoided by setting the FetchAll property. Please see description of the FetchAll property and the CreateConnection option in MyDAC help for more information.

Another alternative that prevents the application from hanging is to switch to the InnoDB storage engine from MyISAM (FetchAll stays False). An application may hang because MyISAM tables can get locked in a read/write collision. If you try to update a table that is not fetched out, MySQL blocks the thread and waits until the table is completely fetched. For details please refer to the MySQL Reference Manual, the [Locking Issues](#) section.

## 8.

### I get an error when opening a Stored Procedure that returns a result set.

Probably this is a bug of the MySQL Server protocol with prepared stored procedures that return record sets. It occurs in the following cases:

- After a call to the Prepare method of MyStoredProc, if the latter had already prepared and opened. The following piece of code demonstrates the problem:

```
MyStoredProc.Prepare;  
MyStoredProc.Open;  
MyStoredProc.UnPrepare;  
MyStoredProc.Prepare;
```

- After a call to the MyStoredProc.Execute method, if the stored procedure returns more than one record set.

## 16 Using MyDAC

### 16.1 Updating Data with MyDAC Dataset Components

MyDAC components that are descendants from [TCustomDADataset](#) provide different means for reflecting local changes to the server.

The first approach is to use automatic generation of update SQL statements. Using this approach you should provide a SELECT statement, everything else will be made by MyDAC automatically. In case when a SELECT statement uses multiple tables, you can use [UpdatingTable](#) property to specify which table will be updated. If [UpdatingTable](#) is blank, the table, that corresponds to the first field in the dataset, is used. This approach is the most preferable and is used in most cases.

Another approach is to set update SQL statements using [SQLInsert](#), [SQLUpdate](#) and [SQLDelete](#) properties. 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. This is useful when there is no possibility to generate correct statement or you need to execute some specific statements. For example, update operations should be made with stored procedure calls.

You may also assign P:Devart.MyDac.TCustomMyDataSet.UpdateObject property with the [TMyUpdateSQL](#) class instance which holds all updating SQL statements in one place. You can generate all these SQL statements using MyDAC design time editors. For more careful customization of data update operations you can use [InsertObject](#), [ModifyObject](#) and [DeleteObject](#) properties of [TMyUpdateSQL](#) component.

#### **See Also**

- [TMyQuery](#)
- [TMyStoredProc](#)
- [TMyTable](#)
- [TMyUpdateSQL](#)

## 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. Lets examine how MyDAC 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 foregin key Dept No that binds "Employee" to "Department".

It is necessary to display and edit these tables.

MyDAC provides two ways to bind tables. First code example shows how to bind two TCustomMyDataSet components (TMyQuery or TMyTable) into MD relationship via parameters.

```
procedure TForm1.Form1Create(Sender: TObject);
var
  Master, Detail: TMyQuery;
  MasterSource: TDataSource;
begin
  // create master dataset
  Master := TMyQuery.Create(Self);
  Master.SQL.Text := 'SELECT * FROM Department';
  // create detail dataset
  Detail := TMyQuery.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.

Now suppose that detail table "Department" foregin 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: TMyQuery;
  MasterSource: TDataSource;
begin
  // create master dataset
  Master := TMyQuery.Create(Self);
  Master.SQL.Text := 'SELECT * FROM Department';
  // create detail dataset
  Detail := TMyQuery.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](#)

## 16.3 Migration from BDE

In MyDAC the interests of BDE application developers were taken into consideration. So starting to use MyDAC after working with BDE would be easy even for developing complex projects. Moreover, MyDAC does not have problems like ones with LiveQuery and compatibility of applications developed using different versions in BDE.

Abandoning BDE gives one more important advantage - positive effect on performance. Instead of complex BDE-ODBC drivers system it uses the fastest access - directly to MySQL server. Also access to MySQL Embedded server is supported.

MyDAC provides special Wizard to simplify the conversion of already existing projects. This Wizard replaces BDE-components in the specified project (dfm-and pas-files) to MyDAC. BDE-components that will be replaced:

- TDatabase -> TMyConnection
- TQuery -> TMyQuery
- TTable -> TMyTable
- TUpdateSQL -> TMyUpdateSQL

To run the Wizard, select BDE Migration Wizard item in MySQL menu and follow the instructions. BDE Migration Wizard does not support C++Builder and Kylix IDEs.

**Note:** The Wizard serves only to simplify routine operations and after the conversion project might be uncompiled.

Below is the list of properties and methods which cannot be converted automatically. Here you can find hints for users to simplify manual replacement.

### TDatabase

- AliasName - specific BDE property. Not supported by MyDAC
- DatabaseName - has a different meaning in BDE and MyDAC. At MyDAC it means MySQL Server database. See [TMyConnection.Database](#) for details
- Locale - see [TMyConnection.Options.CharSet](#)
- KeepConnection - not supported by MyDAC
- Params - see [TMyConnection](#) properties
- Session, SessionAlias, SessionName - MyDAC does not need global management of a group of database connections in an application. So these properties are not supported
- Temporary - has no sense in MyDAC. Additional connections are created but not available for the user. See [TCustomMyDataSet.FetchAll](#) = False for details
- TraceFlags - see [TCustomDASQLMonitor.TraceFlags](#)
- TransIsolation - see [IsolationLevel](#)
- Execute - use [ExecSQL](#) instead of this method
- FlushSchemaCache - not supported by MyDAC
- GetFieldNames - not supported by MyDAC
- IsSQLBased - not supported by MyDAC. For MySQL must be always True
- ApplyUpdates - parameters are not supported. To update only specified DataSets use [TMemDataset.ApplyUpdates](#). Update is performed within a transaction.

### TBDEDataSet

- BlockReadSize - see [TCustomDADataset.FetchRows](#)
- CacheBlobs - MySQL Server does not provide service of suspended BLOB loading
- KeySize - specific BDE property. Not supported by MyDAC.

### TDBDataSet

- AutoRefresh - supported through [TCustomDADataset.RefreshOptions](#)
- DBFlags, DBHandle, DBLocate, DBSession, Handle - BDE-specific property. Not supported by MyDAC
- SessionName - not supported by MyDAC
- UpdateMode - not supported by MyDAC. By default, the behaviour corresponds upWhereKeyOnly. To change this behaviour see [TCustomDADataset.SQLUpdate](#), [TCustomDADataset.SQLDelete](#), [TCustomDADataset.SQLRefresh](#), and [TCustomMyDataSet.Options.CheckRowVersion](#).

### TQuery

- Constrained - specific BDE property. Not supported by MyDAC
- DataSource - see [TCustomDADataset.MasterSource](#)
- Local - specific BDE property. Not supported by MyDAC
- RequestLive - almost all query result sets are updatable. See [TMyQuery.UpdatingTable](#), [TCustomDADataset.ReadOnly](#), CanModify, [TCustomDADataset.SQLInsert](#), [TCustomDADataset](#).

[SQLUpdate](#), [TCustomDADataset.SQLDelete](#).

- Text - specific BDE property. Not supported by MyDAC.

#### **TTable**

- DefaultIndex - not used in MyDAC. If you need to sort a table by any field see P:Devart.MyDac. TCustomMyTable.OrderFields, [TMemDataSet.IndexFieldNames](#)
- Exists, CreateTable, AddIndex, DeleteIndex, StoreDefs, Deletetable, TableType - MyDAC does not allow to create tables by using TTable. If you need to create a table execute 'CREATE TABLE ...' query or use any special third-party tools.
- IndexDefs - not used in MyDAC, but fills on first call
- IndexFieldNames - a list of fields for local sorting. See [TMemDataSet.IndexFieldNames](#)
- IndexFieldCount, IndexFields, IndexFiles, IndexName, GetIndexNames, GetIndexInfo - Not supported by MyDAC
- KeyExclusive - not supported by MyDAC. Use SELECT ... FROM .. WHERE ... to get requested result
- KeyFieldCount - not supported by MyDAC, as key fields are not used for searching on the client side
- TableLevel - specific BDE property. Not supported by MyDAC
- ApplyRange, CancelRange, EditRangeStart, EditRangeEnd, SetRange - MyDAC does not support Range
- BatchMove - has no meaning in MySQL. Use INSERT ... INTO ... SELECT syntax to copy records onto server side
- FindKey, FindNearest, GotoCurrent, GotoKey, GotoNearest, EditKey, SetKey - use [TMemDataSet.Locate](#) and [TMemDataSet.LocateEx](#)
- GetDetailLinkFields - use [TCustomDADataset.DetailFields](#), [TCustomDADataset.MasterFields](#)
- RenameTable - use 'RENAME TABLE ...' script
- ConstraintCallBack, ConstraintsDisabled, DisableConstraints, EnableConstraints - has no meaning in MySQL
- FlushBuffers - see [TMyServerControl.Flush](#)
- Translate - use AnsiToNative and similar functions.

#### **TSession**

MyDAC does not need global management of a group of database connections in an application.

#### **TUpdateSQL**

A complete analogue to [TMyUpdateSQL](#).

---

## 16.4 Secure Connections

Session security depends on several factors, including whether the connection to the host is a trusted connection. If it is not, confidential information can not be transmitted through this connection. MyDAC supports two different ways to increase connection security. They are SSH and SSL. Both SSH and SSL can be implemented with SecureBridge components.

Devart SecureBridge is a non visual component library that provides functionality for SSH tunneling and SSL connections. Usage of SecureBridge is the handiest and fastest way to ensure protected connection to MySQL server. You can read more about SecureBridge at the [SecureBridge home page](#). The detailed step-by-step instructions on setting up SecureBridge you will find in the SecureBridge documentation.

### 1. SSH using SecureBridge

SecureBridge allows you to embed functionality of an SSH client into your application. The following sequence of steps describes how to protect your connection to MySQL server through an SSH tunnel with SecureBridge:

- configure your SSH server like described in the server documentation, or use SecureBridge to make your own SSH server. SecureBridge includes a demo project that implements functionality of an SSH server;
- place the TScSSHClient component of SecureBridge onto your form;
- setup TScSSHClient (assign host name, SSH server port, user name, password) to connect to the SSH server and check the connection;
- place the TMySSHIOHandler component onto your form. This component is included into MyDAC as a [demo project](#);
- place the [TMyConnection](#) component onto your form, and link to its [IOHandler](#) property the instance of TMySSHIOHandler added on the previous step;
- setup [TMyConnection](#) to connect to MySQL server and check the connection.

Now you have an encrypted connection between MySQL server and your application.

### 2. SSH using OpenSSH or other third-party SSH tunnel

SSH works by "Port forwarding" principle and serves to encrypt transferred data.

The following is the step-by-step sequence of actions for the easiest case of using OpenSSH for Windows. The detailed description of each command you can see in the documentation for OpenSSH.

1. Download OpenSSH for Windows from <http://www.sourceforge.net/projects/sshwindows/>

2. Install SSH server

Choose a machine that will be used as SSH server. It does not have to be the same machine that is a MySQL server, but communication channel between SSH server and MySQL server must be protected

Using Windows Control Panel create a user and set a password for him. For example, SSHUser with password SSHPass

Install Open SSH. It is enough to install only Server components

Open OpenSSH/bin folder

Add SSHUser to the list of allowed users:

```
mkpasswd -l -u SSHUser >> ..\etc\passwd
```

Use mkgroup to create a group permissions file

```
mkgroup -l >> ..\etc\group
```

Run OpenSSH service

```
net start opensshd
```

3. Install SSH client

Choose a machine that will be used as SSH client. It does not have to be the same machine where client application (MySQL client) is running, but communication channel between SSH client and MySQL client must be protected

Install Open SSH to SSH client. You may not install server components

Run SSH client

```
ssh.exe -L <SSH port>:<MySQL server>:<MySQL server port> <SSHUser>@<SSH server>
```

<SSH port> - port number of SSH client that will be redirected to the corresponding port of MySQL server

<MySQL server> - name or IP address of the machine where MySQL server is installed

<MySQL server port> - number of MySQL server port. As usual, 3306.

<SSHUser> - user name created in p. 2

<SSH server> - name or IP address of the machine where SSH server is installed in p.

2

For example,

```
ssh.exe -L 3307:server:3306 SSHUser@192.168.0.116
```

At the first start you will be suggested to confirm a connection with the specified SSH server. Enter "yes" for confirmation.

On each start of SSH you must enter a password set in p. 2

#### 4. Configure TMyConnection

```
MyConnection1.Server := <SSH client>;
```

```
MyConnection1.Port := <SSH port>;
```

If SSH client was installed at the same machine as MySQL client, you can assign 'localhost' to MyConnection1.Server.

Pay attention that in the specified sequence above check of SSHUser authentication is performed by Windows. About the methods of higher protection (key authentication etc) see documentation for OpenSSH.

To get more detailed information on using encrypted connections refer to [MySQL Reference Manual](#).

### 3. SSL using SecureBridge

SecureBridge also allows you to embed functionality of an SSL client into your application. The following sequence of steps describes how to protect your connection to MySQL server with SSL using SecureBridge:

- Place the TMySSLIOHandler component onto the form.
- Select a storage object in the Storage property. More information about storage setup you will find in the SSL client setup topic of SecureBridge help.
- Specify the server certificate in the CACertName property.
- Specify the client certificate in the CertName property.
- Place the TMyConnection component onto the form and setup it to connect to the MySQL server.
- Assign the TMySSLIOHandler object to the IOHandler property of TMyConnection.
- Connect to MySQL server by setting TMyConnection.Connected to True.

### 4. SSL

SSL is based on algorithms of asymmetric encryption and digital signature. Consult MySQL Reference Manual for information on how to [enable SSL support for MySQL server](#) and [generate certificates](#).

**Note** that usage of SSL is more preferable for MySQL connections than SSH because of less required settings and higher performance.

#### See Also

- [SecureBridge home page](#)
- [IOHandler](#)
- [SSLOptions](#)



## 16.5 Network Tunneling

Usually when a client needs to connect to server it is assumed that direct connection can be established. Nowadays though, due to security reasons or network topology, it is often necessary to use a proxy or bypass a firewall. This article describes different ways to connect to MySQL server with MyDAC.

- [Direct connection](#)
- [Connection through HTTP tunnel](#)
  - [Connection through proxy and HTTP tunnel](#)
- [Additional information](#)

### Direct connection

Direct connection to server means that server host is accessible from client without extra routing and forwarding. This is the simplest case. The only network setting you need is the host name and port number. This is also the fastest and most reliable way of communicating with server. Use it whenever possible.

The following code illustrates the simplicity:

```
MyConnection := TMyConnection.Create(self);
MyConnection.Server := 'localhost';
MyConnection.Port := 3306;
MyConnection.Username := 'root';
MyConnection.Password := 'root';
MyConnection.Connect;
```

### Connection through HTTP tunnel

Sometimes client machines are shielded by a firewall that does not allow you to connect to server directly at the specified port. If the firewall allows HTTP connections, you can use MyDAC together with HTTP tunneling software to connect to MySQL server.

MyDAC supports HTTP tunneling based on the PHP script.

An example of the web script tunneling usage can be the following: you have a remote website, and access to its database through the port of the database server is forbidden. Only access through HTTP port 80 is allowed, and you need to access the database from a remote computer, like when using usual direct connection.

You need to deploy the tunnel.php script, which is included into the provider package on the web server. It allows access to the database server to use HTTP tunneling. The script must be available through the HTTP protocol. You can verify if it is accessible with a web browser. The script can be found in the HTTP subfolder of the installed provider folder, e. g. %Program Files%\Devart\MyDac for Delphi X\HTTP\tunnel.php. The only requirement to the server is PHP 5 support.

To connect to the database, you should set TMyConnection parameters for usual direct connection, which will be established from the web server side, the Options.Protocol property to mpHttp, and set the following parameters, specific for the HTTP tunneling:

| Property                                   | Mandatory | Meaning   |
|--|-----------|---|
| HttpOptions.Url                            | Yes       | Url of the tunneling PHP script. For example, if the script is in the server root, the url can be the following: http://localhost/tunnel.php.             |
| HttpOptions.Username, HttpOptions.Password | No        | Set this properties if the access to the website folder with the script is available only for registered users authenticated with user name and password. |

### Connection through proxy and HTTP tunnel

Consider the previous case with one more complication.

HTTP tunneling server is not directly accessible from client machine. For example, client address is 10.0.0.2, server address is 192.168.0.10, and the MySQL server listens on port 3307. The client and server reside in different networks, so the client can reach it only through proxy at address 10.0.0.1, which listens on port 808. In this case in addition to the TMyConnection.HttpOptions options you have to setup a HttpOptions.ProxyOptions object as follows:

```
MyConnection := TMyConnection.Create(self);
MyConnection.Server := '192.168.0.10';
MyConnection.Port := 3307;
MyConnection.Username := 'root';
```

```
MyConnection.Password := 'root';  
MyConnection.Options.Protocol := mpHttp;  
MyConnection.HttpOptions.Url := 'http://server/tunnel.php';  
MyConnection.HttpOptions.ProxyOptions.Hostname := '10.0.0.1';  
MyConnection.HttpOptions.ProxyOptions.Port := 808;  
MyConnection.HttpOptions.ProxyOptions.Username := 'ProxyUser';  
MyConnection.HttpOptions.ProxyOptions.Password := 'ProxyPassword';  
MyConnection.Connect;
```

Note that setting parameters of `MyConnection.HttpOptions.ProxyOptions` automatically enables proxy server usage.

### **Additional information**

Technically speaking, there is one more way to tunnel network traffic. The Secure Shell forwarding, or SSH, can be used for forwarding data. However, main purpose of SSH is traffic encryption rather than avoiding firewalls or network configuration problems. The Secure Connections article describes how to use SSH protocol in MyDAC.

Keep in mind that traffic tunneling or encryption always increase CPU usage and network load. It is recommended that you use direct connection whenever possible.

---

©

1997-2012 Devart. All Rights Reserved.

## 16.6 Embedded Server

Since version 4.0 MySQL server supports Embedded server. Embedded server is an easy to install server used by applications that do not require multi-user work with MySQL server. For example, Embedded server can be used for money access machines, automatic cash desks, different electronic facilities and so on. Please refer to [MySQL Reference Manual](#) for more details on features and using of Embedded server. Also you can find some information about licensing Embedded server in [MySQL Reference Manual](#). Please refer to Embedded Demo for a sample.

### Which version of Embedded server to use

MySQL Embedded Server 4.0 should be recompiled to be used in your application.

MySQL Embedded Server 5.0 is not included into the binary installation pack. Below is a quotation from the [MySQL Reference Manual](#):

*"The Embedded MySQL server library is NOT part of MySQL 5.0. It is part of previous editions and will be included in future versions, starting with MySQL 5.1."*

That is why we have not tested MyDAC with the MySQL Embedded Server 5.0.

So, we recommend using MySQL Embedded Server 4.1. As MySQL Embedded Server has some problems working with the InnoDB storage, we recommend disabling this storage engine. You can do this by checking the "Disable InnoDB storage engine" option in the TMyEmbConnection editor on the Params tab. Another way is to add the --skip-innodb parameter to the TMyEmbConnection component manually.

### Installation

- Windows

Copy libmysqld.dll file to the folder available for executable file of the application. Please see a detailed description of accessible paths at LoadLibrary description

A typical structure of folders for an application using Embedded Server:

*Project.exe* - executable file of your application

*libmysqld.dll* - MySQL Embedded server library

*share/english/errmsg.sys* - file with MySQL Embedded server messages

*data/* - data directory ([DataDir](#)). See a structure of this folder in [MySQL Reference Manual](#)

*data/mysql/* - directory with service data of MySQL (user access rights, and so on) *data/Database/* - directory with user data. See [TCustomMyConnection.Database](#)

- Linux

- Copy libmysqld.so.14.0.0 file to /usr/lib folder

- At /usr/lib folder execute the following commands to create links:

*In libmysqld.so.14.0.0 libmysqld.so*

*In libmysqld.so.14.0.0 libmysqld.so.14*

- Copy files needed for working of Embedded Server. As a rule, it is error message file, for example *share/english/errmsg.sys*.
- Create a folder for data
- If it is necessary, copy files with data to the data folder

### Settings

On the start (first opening a connection), MySQL Embedded Server searches for the setting values in the next order:

- [TMyEmbConnection.Params](#)
- [<Application exe-file name (with extension)> section of configuration file (my.ini or my.cnf) - settings specific to particular application.
- [Embedded section of configuration file - settings specific to Embedded server
- [Server section of configuration file - common settings for MySQL server and Embedded server.

Usually to set-up Embedded Server it is enough to set basedir and datadir. But sometimes some additional settings are required, for example to disable using InnoDB engine (--skip-innodb). The detailed list of settings you can find at [MySQL Reference Manual](#).

Pay attention that all paths must be set through "/" but not "\".

Note, parameters names are case-sensitive.

If datadir is located in the read-only storage, then you need to set [OnLog](#) and OnLogError event handlers to prevent server from attempts to create log-files in datadir.

### Limitations

Simultaneous access to the same data from several instances of MySQL server (for example, to MySQL server and Embedded server) can be a reason of data loss.

**See Also**

- Embedded Demo
  - [TMyEmbConnection](#)
  - [TMyEmbConnection.Params](#)
  - [TMyConnection.Options.Embedded](#)
-

## **16.7 National Characters**

On transferring data between client and server sides, server must know the encoding format used at the client. You can set the coding using server means by assigning corresponding parameters at the server settings (see MySQL Reference Manual for details), or by client methods setting [TCustomMyConnection.Options.Charset](#) or [TCustomMyConnection.Options.UseUnicode](#) properties. The first way is less suitable as it requires meddling in server settings that is not always possible. The second way is more convenient but it can cause insignificant delay while establishing a connection.

Let us see the specific of using Charset and UseUnicode options. These options are mutually exclusive, thus on setting UseUnicode property to True a value of Charset will be ignored.

By default, Charset = "", and UseUnicode = False. And the server makes conversions according to its settings.

If Charset property is enabled, then on establishing a connection "SET NAMES <Charset>" query is automatically passed to the server to explicitly notify the server about the character set of the client. To get a list of available charsets, you can execute "SHOW CHARSET" query. Pay attention that on setting Charset = 'utf8' values of all string fields will be converted to this encoding format that in most cases can make impossible to use DataAware components.

Setting UseUnicode property to True allows to retrieve string data at the client side in Unicode encoding format that let you work simultaneously almost with all languages. All TStringField values will be converted to TWideStringField. This behaviour is suitable, for example, when creating a database of books in the library, when next to the name of a book you should also store its name in the original language. Please note that setting this option has some imperfections. Firstly, all string data at the client side will be converted, and it can cause a delay in working. Secondly, standard Borland Data-aware controls do not support Unicode (Wide-strings) and you have to use third-party components.

### **See Also**

- [TCustomMyConnection.Options](#)

## 16.8 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
```

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 [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 active transactions;
- 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), MyDAC 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](#)

## 16.9 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 authentication. 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
```

These settings minimize the number of requests to the server.

### Disconnected mode features

If you perform a query with the [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](#)
- [FetchAll](#)
- [Devart.MyDac.TMyQuery.LockMode](#)
- [TCustomDAConnection.Pooling](#)
- [TCustomDAConnection.Connect](#)
- [TCustomDAConnection.Disconnect](#)
- [Working in unstable network](#)

## 16.10 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, MyDAC 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 a MySQL database:

```
CREATE TABLE DECIMAL_TYPES
(
  ID INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  VALUE1 DECIMAL(4, 0),
  VALUE2 DECIMAL(10, 0),
  VALUE3 DECIMAL(15, 0),
  VALUE4 DECIMAL(5, 2),
  VALUE5 DECIMAL(10, 4),
  VALUE6 DECIMAL(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 TIntegerField, 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:

| MySQL data type | Default Delphi field type | Destination Delphi field type |
|-----------------|---------------------------|-------------------------------|
| DECIMAL(4,0)    | ftFloat                   | ftSmallint                    |
| DECIMAL(10,0)   | ftFloat                   | ftInteger                     |
| DECIMAL(15,0)   | ftFloat                   | ftLargeint                    |
| DECIMAL(5,2)    | ftFloat                   | ftFloat                       |
| DECIMAL(10,4)   | ftFloat                   | ftBCD                         |
| DECIMAL(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      := myDecimal;
  MinPrecision := 0;
  MaxPrecision := 4;
  MinScale    := 0;
  MaxScale    := 0;
  FieldType   := ftSmallint;
  MyConnection.DataTypeMap.AddDBTypeRule(DBType, MinPrecision, MaxPrecision, MinScale, MaxScale,
  end;
```

This is an example of the detailed rule setting, and it is made for maximum visualization. Usually, rules



are set much shorter, e.g. as follows:

```
// clear existing rules
MyConnection.DataTypeMap.Clear;
// rule for DECIMAL(4,0)
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 0, 4, 0, 0, ftSmallint);
// rule for DECIMAL(10,0)
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 5, 10, 0, 0, ftInteger);
// rule for DECIMAL(15,0)
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 11, rlAny, 0, 0, ftLargeint);
// rule for DECIMAL(5,2)
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 0, 9, 1, rlAny, ftFloat);
// rule for DECIMAL(10,4)
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 10, rlAny, 1, 4, ftBCD);
// rule for DECIMAL(15,6)
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 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 MySQL database:

```
CREATE TABLE DECIMAL_TYPES
(
  ID INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  VALUE1 DECIMAL(5, 2),
  VALUE2 DECIMAL(10, 4),
  VALUE3 DECIMAL(15, 6)
)
```

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

| MySQL data type | Default Delphi field type | Destination field type |
|-----------------|---------------------------|------------------------|
| DECIMAL(5,2)    | ftFloat                   | ftFloat                |
| DECIMAL(10,4)   | ftFloat                   | ftBCD                  |
| DECIMAL(15,6)   | ftFloat                   | ftFMTBCD               |

If rules are set in the following way:

```
MyConnection.DataTypeMap.Clear;
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 0, 9, rlAny, rlAny, ftFloat);
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 0, rlAny, 0, 4, ftBCD);
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 0, rlAny, 0, rlAny, ftFMTBCD);
```

it will lead to the following result:

| MySQL data type | Delphi field type |
|-----------------|-------------------|
| DECIMAL(5,2)    | ftFloat           |
| DECIMAL(10,4)   | ftBCD             |
| DECIMAL(15,6)   | ftFMTBCD          |

But if rules are set in the following way:

```
MyConnection.DataTypeMap.Clear;
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 0, rlAny, 0, rlAny, ftFMTBCD);
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 0, rlAny, 0, 4, ftBCD);
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 0, 9, rlAny, rlAny, ftFloat);
```

it will lead to the following result:

| MySQL data type | Delphi field type |
|-----------------|-------------------|
| DECIMAL(5,2)    | ftFMTBCD          |
| DECIMAL(10,4)   | ftFMTBCD          |
| DECIMAL(15,6)   | ftFMTBCD          |

This happens because the rule

```
MyConnection.DataTypeMap.AddDBTypeRule(myDecimal, 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 INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    FIRSTNAME VARCHAR(20),
    LASTNAME VARCHAR(30),
    GENDER_CODE VARCHAR(1),
    BIRTH_DTTM DATETIME
)
```

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:

```
MyConnection.DataTypeMap.Clear;
MyConnection.DataTypeMap.AddDBTypeRule(myDateTime, 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:

```
MyQuery.DataTypeMap.Clear;
MyQuery.DataTypeMap.AddDBTypeRule(myDateTime, 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:

```
MyConnection.DataTypeMap.Clear;
MyConnection.DataTypeMap.AddDBTypeRule(myDateTime, ftDate);
MyQuery.DataTypeMap.Clear;
MyQuery.DataTypeMap.AddDBTypeRule(myDateTime, ftDateTime);
```

In this case, in all DataSets for the DATETIME type fields with the ftDate type will be created, and for MyQuery - 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 INT(11) NOT NULL AUTO_INCREMENT 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:

```

MyQuery.DataTypeMap.Clear;
MyQuery.DataTypeMap.AddDBTypeRule(myChar, 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:

```

MyQuery.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:

```

MyConnection.DataTypeMap.AddDBTypeRule(myVarchar, 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.

## 16.11 Data Encryption

MyDAC 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 INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ENAME VARBINARY(2000) DEFAULT NULL,
    HIREDATE VARBINARY(200) DEFAULT NULL,
    SAL VARBINARY(200) DEFAULT NULL,
    FOTO BLOB DEFAULT NULL
)
```

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.

```
MyQuery.SQL.Text := 'SELECT * FROM EMP';
```

---

```
MyQuery.Encryption.Encryptor := MyEncryptor;  
MyQuery.Encryption.Fields := 'ENAME, HIREDATE, SAL, FOTO';  
MyEncryptor.Password := '11111';  
MyQuery.DataTypeMap.AddFieldNamedRule ('ENAME', ftString);  
MyQuery.DataTypeMap.AddFieldNamedRule ('HIREDATE', ftDateTime);  
MyQuery.DataTypeMap.AddFieldNamedRule ('SAL', ftFloat);  
MyQuery.Open;
```

---

## 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 `T:Devart.MyDac.TMySQL` component is more preferable than [TMyQuery](#). 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 `P:Devart.Dac.TDADatasetOptions.CompressBlobMode`. You can also use [TMyConnection.Options.Compress](#). Setting [TMyTable.Options.UseHandler](#) can give an additional performance under high server load.

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.

You can also increase the data sending performance a bit (several percents) by using `Dataset.UpdateObject`, `Dataset.ModifyObject`, etc.

Little additional performance improvement can be reached by setting the [AutoPrepare](#) property for these objects.

### Insert

If you are about to insert a large number of records into a table, you should use the [TDevart.MyDac.](#)

---

[TMyLoader](#) component instead of Insert/Post methods, or execution of the INSERT commands multiple times in a cycle. Sometimes usage of [TDevart.MyDac.TMyLoader](#) improves performance several times.

---

© 1997-2012 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](#), [Server](#), [T:Devart.Odac.TOraSession](#), [Password](#), [Database](#), [IsolationLevel](#), [Port](#), [IOHandler](#), [ConnectionTimeout](#), [Compress](#), [Direct](#), [Embedded](#), [Protocol](#), [Charset](#), [UseUnicode](#), [NumericType](#). 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 [TMyConnectionPoolManager](#). 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](#)





## 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 MyDAC 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;
```

MyDAC 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. For example, the following is a script that creates a new user account and grants required privileges.

```
Script1.SQL.Add('CREATE USER &Username IDENTIFIED BY &Password;');  
Script1.SQL.Add('GRANT &Privileges TO &Username;');
```

To execute the script for another user you do not have to change the script SQL property, you can just set required 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](#)
-

## **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:

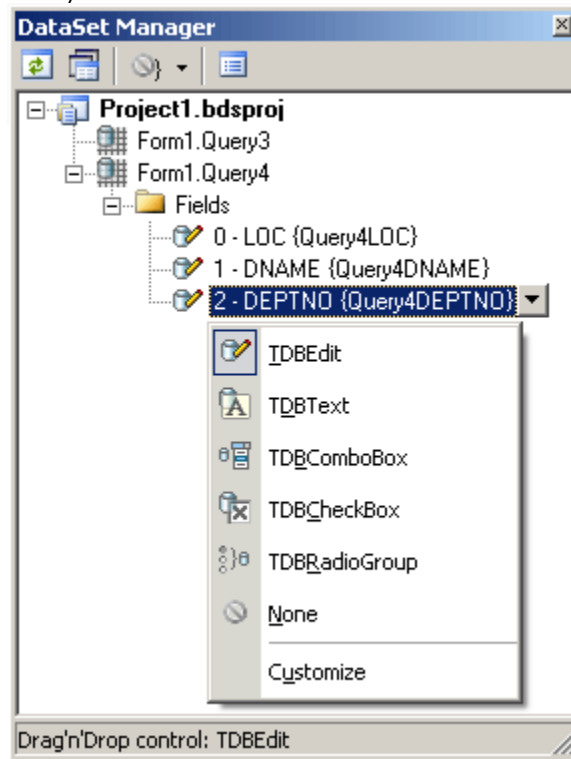
- Devart.Dac.dll
- Devart.Vcl.dll
- Devart.Dac.Design.dll
- Devart.Dac.AdoNet.dll

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.

## 16.16 DataSet Manager

### DataSet Manager window

The DataSet Manager window displays the datasets in your project. You can use the DataSet Manager window to create a user interface (consisting of data-bound controls) by dragging items from the window onto forms in your project. Each item has a drop-down control list where you can select the type of control to create prior to dragging it onto a form. You can customize the control list with additional controls, including the controls you have created.



Using the DataSet Manager window, you can:

- Create forms that display data by dragging items from the DataSet Manager window onto forms.
- Customize the list of controls available for each data type in the DataSet Manager window.
- Choose which control should be created when dragging an item onto a form in your Windows application.
- Create and delete TField objects in the DataSets of your project.

### Opening the DataSet Manager window

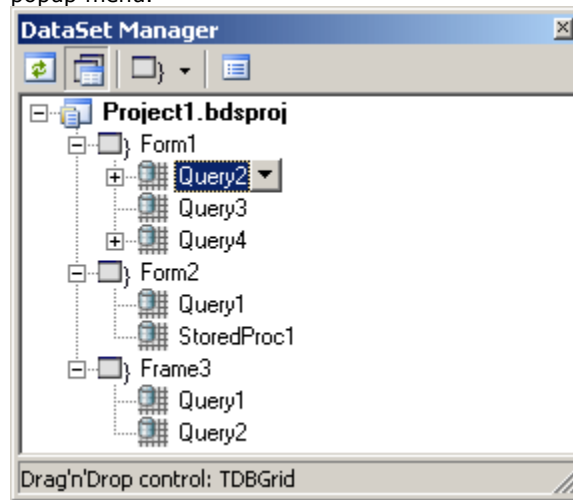
You can display the DataSet Manager window by clicking DataSet Manager on the Tools menu. You can also use IDE desktop saving/loading to save DataSet Manager window position and restore it during the next IDE loads.

### Observing project DataSets in the DataSet Manager Window

By default DataSet Manager shows DataSets of currently open forms. It can also extract DataSets from all forms in the project. To use this, click *Extract DataSets from all forms in project* button. This settings is remembered. Note, that using this mode can slow down opening of the large projects with plenty of forms and DataSets. Opening of such projects can be very slow in Borland Delphi 2005 and Borland Developer Studio 2006 and can take up to several tens of minutes.

DataSets can be grouped by form or connection. To change DataSet grouping click the *Grouping mode* button or click a down. You can also change grouping mode by selecting required mode from the

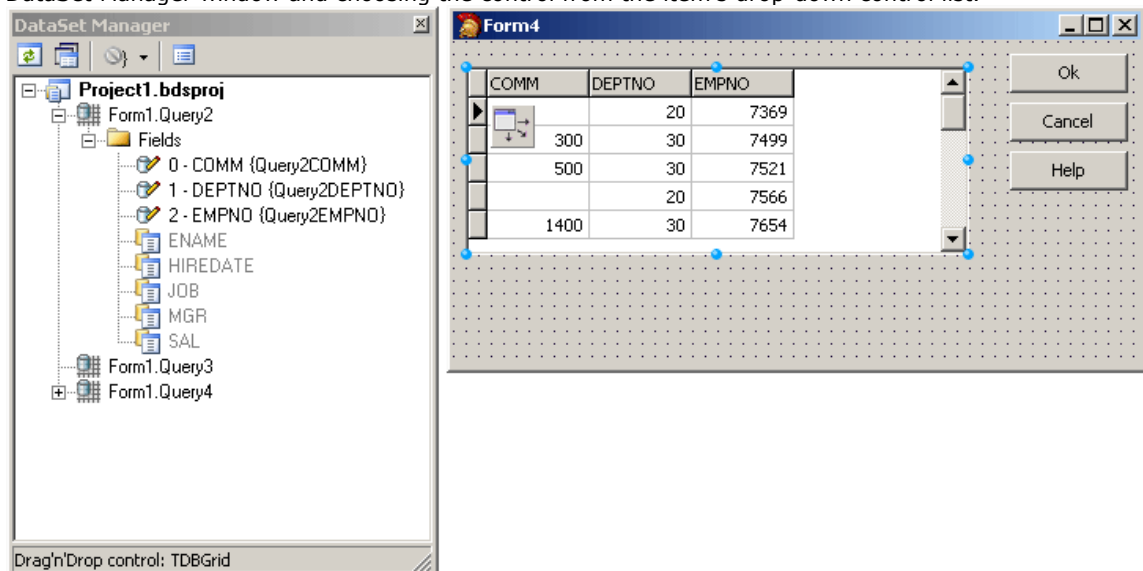
DataSet Manager window popup menu.



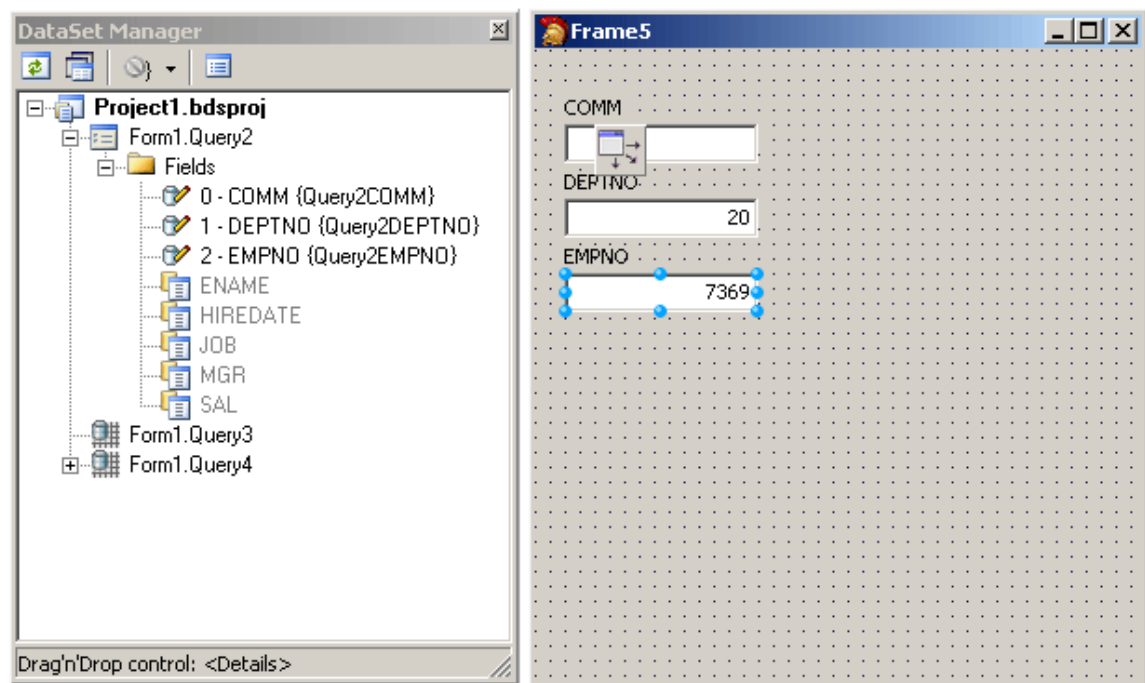
### Creating Data-bound Controls

You can drag an item from the DataSet Manager window onto a form to create a new data-bound control. Each node in the DataSet Manager window allows you to choose the type of control that will be created when you drag it onto a form. You must choose between a Grid layout, where all columns or properties are displayed in a TDataGrid component, or a Details layout, where all columns or properties are displayed in individual controls.

To use grid layout drag the dataset node on the form. By default TDataSource and TDBGrid components are created. You can choose the control to be created prior to dragging by selecting an item in the DataSet Manager window and choosing the control from the item's drop-down control list.

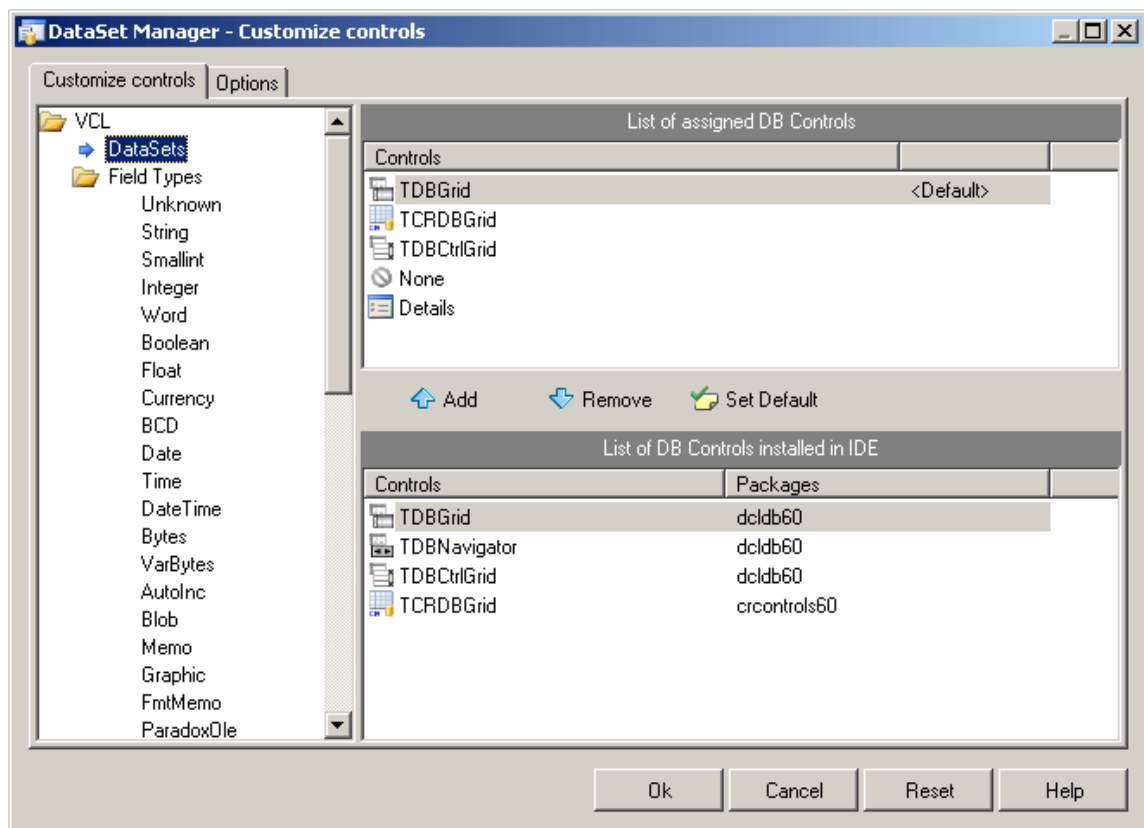


To use Details layout choose Details from the DataSet node drop-down control list in the DataSet Manager window. Then select required controls in the drop-down control list for each DataSet field. DataSet fields must be created. After setting required options you can drag the DataSet to the form from the DataSet window. DataSet Manager will create TDataSource component, and a component and a label for each field.



### Adding custom controls to the DataSet Manager window

To add custom control to the list click the *Options* button on the DataSet Manager toolbar. A *DataSet Manager - Customise controls* dialog will appear. Using this dialog you can set controls for the DataSets and for the DataSet fields of different types. To do it, click DataSets node or the node of field of required type in *DB objects groups* box and use *Add* and *Remove* buttons to set required control list. You can also set default control by selecting it in the list of assigned DB controls and pressing *Default* button.



The default configuration can easily be restored by pressing Reset button in the *DataSet Manager - Options* dialog.

### Working with TField objects

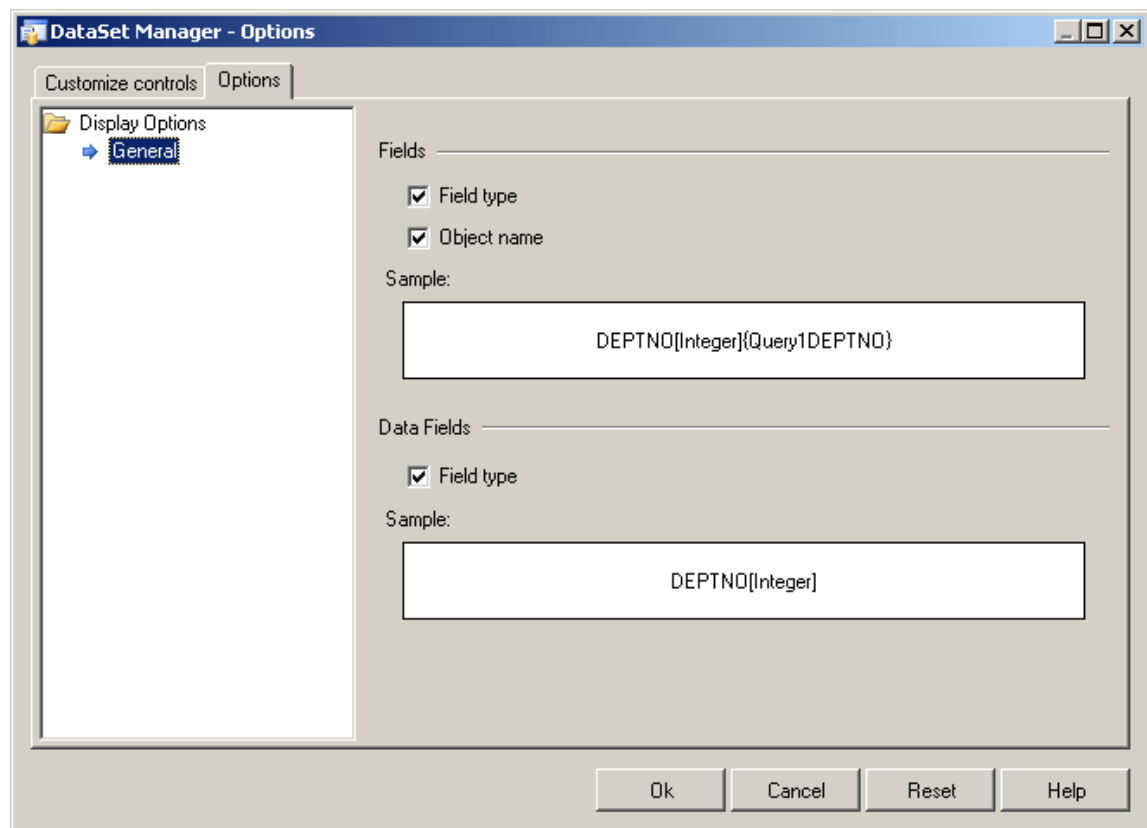
DataSet Manager allows you to create and remove TField objects. DataSet must be active to work with its fields in the DataSet Manager. You can add fields, based on the database table columns, create new fields, remove fields, use drag-n-drop to change fields order.

To create a field based on the database table column right-click the Fields node and select *Create Field* from the popup menu or press <Insert>. Note that after you add at least one field manually, DataSet fields corresponding to data fields will not be generated automatically when you drag the DataSet on the form, and you can not drag such fields on the form. To add all available fields right-click the Fields node and select *Add all fields* from the popup menu.

To create new field right-click the Fields node and select *New Field* from the popup menu or press <Ctrl+Insert>. The New Field dialog box will appear. Enter required values and press OK button.

To delete fields select these fields in the DataSet Manager window and press <Delete>.

DataSet Manager allows you to change view of the fields displayed in the main window. Open the *Customize controls* dialog, and jump to the Options page.



You can choose what information will be added to names of the Field and Data Field objects in the main window of DataSet Manager. Below you can see the example.



## **16.17 DBMonitor**

To extend monitoring capabilities of MyDAC applications there is an additional tool called DBMonitor. It is provided as an alternative to Borland SQL Monitor which is also supported by MyDAC. DBMonitor is an easy-to-use tool to provide visual monitoring of your database applications.

DBMonitor has the following features:

- multiple client processes tracing;
- SQL event filtering (by sender objects);
- SQL parameter and error tracing.

DBMonitor is intended to hamper an application being monitored as little as possible.

To trace your application with DB Monitor you should follow these steps:

- drop [TMySQLMonitor](#) component onto the form;
- turn [moDBMonitor](#) option on;
- set to True the Debug property for components you want to trace;
- start DBMonitor before running your program.

## **16.18 Migration Wizard**

### **NOTE:**

Migration Wizard is available only for Delphi IDE and is not available for C++Builder. BDE Migration Wizard allows you to convert your BDE projects to MyDAC. This wizard replaces BDE components at the specified project (dfm-and pas-files) to MyDAC.

To convert a project, perform the following steps.

- Select **BDE Migration Wizard** from **MySQL** menu
- Select **Replace BDE components** to replace corresponding components with MyDAC 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 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 Migration Wizard** from **MySQL** 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.

### **See Also**

- [Migration from BDE](#)

## **16.19 Writing GUI Applications with MyDAC**

MyDAC 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 MyDacVcl (MyDacClx under Linux) unit in your application. This feature is needed for writing console applications.

### ***D I h and C++B Id***

By default MyDAC does not require Forms, Controls and other GUI related units. Only [TMyConnectDialog](#) component require the Forms unit.

### ***Kyl x***

By default MyDAC does not require QT library. Only T[SPCaps ConnectDialog component includes QT-dependent code.

## 16.20 Compatibility with Previous Versions

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

The [BaseSQL](#) property is similar to the SQL property, but it does not store changes made by [AddWhere](#), [DeleteWhere](#), and [SetOrderBy](#) methods. After assigning an SQL text and modifying it by one of these methods, all subsequent changes of the SQL property will not be reflected in the BaseSQL property. This behavior was changed in MyDAC 4.00.2.8. To restore old behavior, set the BaseSQLOldBehavior variable to True.

### DBAccess.SQLGeneratorCompatibility:

If the manually assigned [RefreshSQL](#) property contains only "WHERE" clause, MyDAC 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 MyDAC 5.00.0.4. To restore the old behavior, set the BaseSQLOldBehavior variable to True.

### MemDS.SendDataSetChangeEventAfterOpen:

Starting with MyDAC 5.20.0.11, 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 MyDAC 5.20.0.12, 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.

### MyClasses. Strings65535ToMemo:

Control flow functions of MySQL (like IF, CASE) change data type of LONGMEMO and LONGBLOB fields. It causes wrong description of these fields by MyDAC and truncating their data. To avoid these problems, MyDAC tries to restore the correct data type. This behaviour was introduced in MyDAC 5.10.0.9. To disable this behaviour, set the Strings65535ToMemo variable to False.

### DBAccess.ParamStringAsAnsiString:

This variable has sense for Delphi 2009 and higher.

Set its value to True to use the AnsiString property when setting the parameter value through TDAParam.AsString. Otherwise the AsWideString property is used. The default value is False.

### DBAccess.RefreshParamsOnInsert:

Starting with MyDAC 5.50.0.36, when master/detail relationship is used on inserting a new record into master table parameters in detail table are not updated. To restore the old behavior, set the RefreshParamsOnInsert variable to True.

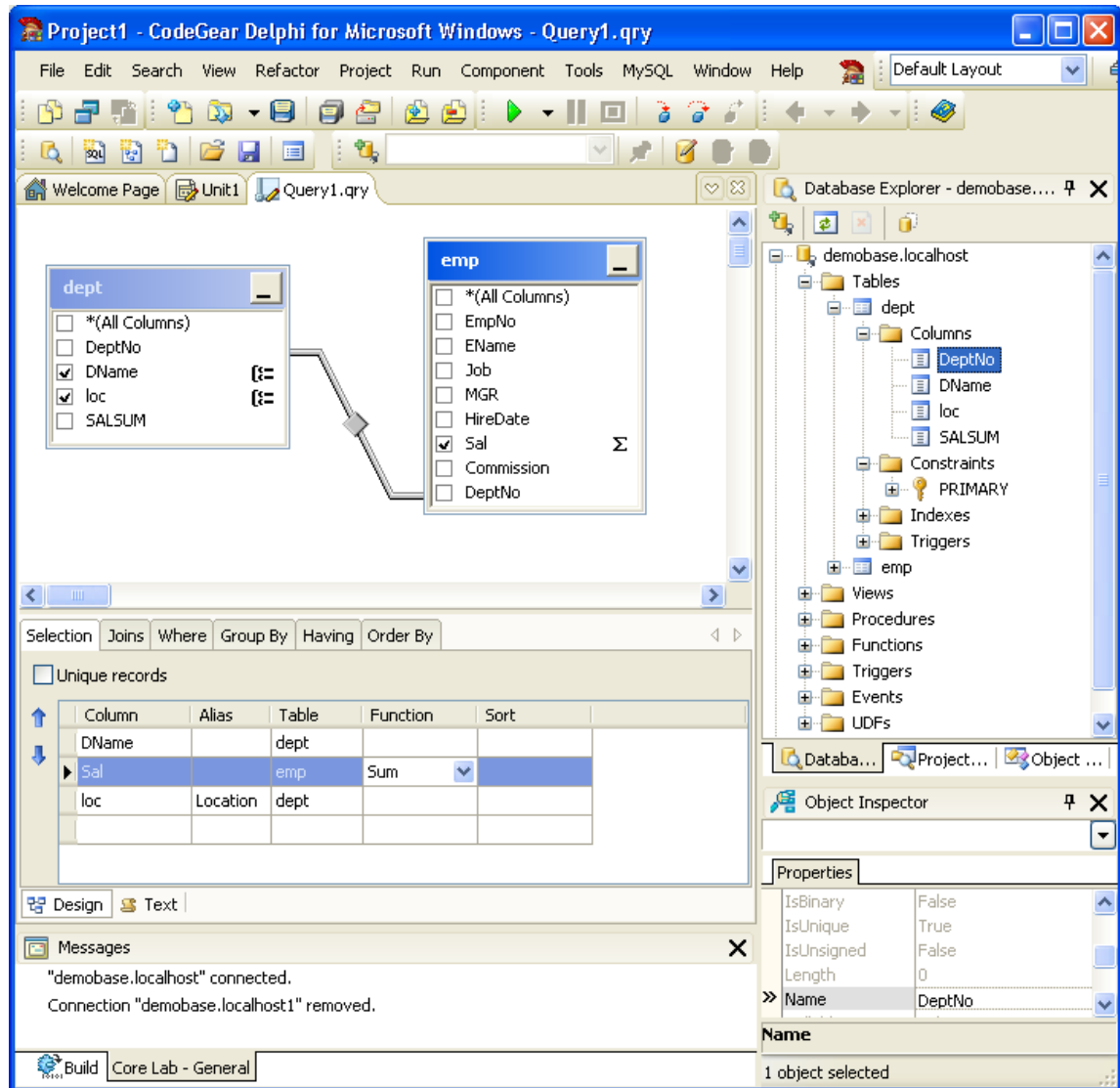
## **16.21 dbForge Fusion for MySQL**

This article provides basic information about dbForge Fusion for MySQL (formerly known as MyDeveloper Tools). The article explains what is dbForge Fusion for MySQL, where to download it, how to install and start using it. For thorough information on dbForge Fusion for MySQL please refer to its own documentation.

### **Introduction**

dbForge Fusion for MySQL is a powerful IDE add-in designed to automate and simplify the MySQL database development process. It integrates into Visual Studio and Delphi, making all database development and administration tasks available from your favorite IDE. Using dbForge Fusion for MySQL, you can:

- Create, modify and delete database connections and easily navigate server-specific database information in tree view
- Create, modify, and drop various database objects
- View and edit table data with an intelligent grid-based editor
- Edit SQL code in a comfortable scripting environment with context-sensitive code completion, syntax highlighting, outlining, code navigation and code templates
- Debug SQL scripts and stored procedures
- Open and save SQL documents
- Create and execute SQL statements
- Examine the SQL query execution plan
- Visually design queries using Query Builder
- Create and deploy MySQL database projects
- Administrate users, and privileges
- Easily export data, database objects and databases
- Create new components by dragging items from Database Explorer
- Take advantage of the extended integration functionality in MyDAC component designers



### Versions and Compatibility

[dbForge Fusion for MySQL](#) is available in two editions.

- [dbForge Fusion for Visual Studio](#), that includes support for Visual Studio .NET 2005 and Visual Studio 2008
- [dbForge Fusion for Delphi](#), that includes support for Delphi and C++Builder 2009, and CodeGear RAD Studio 2007

MyDAC 5.55 is compatible with dbForge Fusion for MySQL 3.00. If you are using MyDAC starting with version 5.00 up to 5.55, you can install MyDeveloper Tools for Delphi 2.00 and higher.

### Related Products

Devart also offers a number of other database products, including dbForge Studio, the standalone version of this MySQL development tool, and OraDeveloper Tools and OraDeveloper Studio, a parallel product line for Oracle.

You can find a full description of all the Devart database tools on the [Devart web site](#).

### Downloading and Installing

dbForge Fusion for MySQL comes in separate installation packages for each supported IDE. If you have purchased MyDAC Developer Edition, you are entitled to receive one free license for the full version of dbForge Fusion for MySQL. Please consult your order confirmation email for the instructions on how to download the installation package for the IDE you are using. Otherwise, you can purchase dbForge

Fusion for MySQL on the [Devart website](#) or download a free trial copy of the version you need from the MyDeveloperTools [download page](#).

Before installing dbForge Fusion for MySQL, make sure that no older versions of the software are installed on the target IDE. Close all IDE instances, launch the downloaded installer, and follow the instructions of the wizard to install the product. Now upon launching the IDE, the dbForge Fusion for MySQL logo should appear on the splash screen and a new dbForge Fusion for MySQL toolbar should be added to the IDE interface.

## Basic Usage Instructions

### Working with database connections

To start using dbForge Fusion for MySQL, you will need to establish a connection to the database you want to work with first. After a connection is established, you can open it to retrieve and manipulate the data provided.

In dbForge Fusion for MySQL database connections are managed in a separate Database Explorer window. The Database Explorer window displays all available database connections at the top level of its tree hierarchy.

To add a database connection in the Database Explorer, complete the following steps.

1. On the Database Explorer window toolbar, press the New Connection button or select the appropriate item from the popup menu.
2. On the "Data Source" tab of the Database Connection Properties dialog box, choose a database server from the list.
3. On the "Connection" tab of the Database Connection Properties dialog box, provide the main logon information required to connect to the server.
4. On the "Parameter" tab of the Database Connection Properties dialog box, provide all specific connection properties you need.
5. Test the connection you've created by clicking the "Test Connection" button.
6. Click OK to establish the database connection.

The Database Connection Properties dialog box will close, and a newly created database connection will appear at the top level of the tree, allowing you to access your MySQL database.

You can modify an existing database connection by right-clicking on its node in Database Explorer, and choosing "Modify Connection" from the node popup menu. In the Database Connection Properties dialog box that will appear make any necessary changes to the connection properties. After you apply these changes by pressing OK, the database connection will close and reopen with the new parameters.

You can rename database connection using the in-place item editor of the Database Explorer tree view.

You can drop a database connection by choosing "Delete" from its node popup menu.

### Displaying server-specific database information in tree view

After a database connection is created and opened, you can explore its database objects by navigating its hierarchy tree. Database Explorer allows you to view, edit, create and drop database objects for all connections. To modify or add an item to the database schema, right click on its node to display a popup menu with the available actions for this node.

If other users are modifying this database simultaneously, you can update the list of database objects displayed in the Database Explorer and their properties to reflect the latest changes by pressing the "Refresh" button.

### Working with database objects

You can create database objects by using the Database Explorer popup menu or by pressing the "Create New Database Object" button on the dbForge Fusion for MySQL toolbar.

To modify an object displayed in the Database Explorer tree, double click on its node to invoke its object editor. In dbForge Fusion for MySQL, objects are represented as tabbed documents that appear in the main IDE editor space. Object editor documents have several interrelated views, and let you apply or cancel the changes you make manually.

Object's properties can be also quickly viewed in a separate Properties window by navigating to that object in the Database Explorer.

To drop a database object, select it and choose "Delete" from popup menu.

### Working with database projects

You can use database projects to manage SQL scripts, query files, and database objects easily. Database projects let you organize related scripts and queries and provide fast access to the selected database objects. They can be created, compiled, and deployed. Some of the advanced benefits of using database projects include the possibility of automatic compilation of a collection of source objects, creation of a whole database from several scripts, and specification of the project deployment order. Projects are an added feature of dbForge Fusion for MySQL, and project folder and file structure, connection and database object links, deployment order are stored locally in a file with the .mysqldev extension.

To create a new project select Tools Devart Developer Tools MySQL New Blank Project.

To open an existing project, select Tools Devart Developer Tools Open Project ...

Each project can be associated with one connection. Project deployment is performed through this connection. To associate a connection with the project, right click on the connection in Database Explorer and select "Assign to project" from the popup menu.

To deploy a project perform the following steps.

1. Select Tools Devart Developer Tools Project Deployment Order.
2. Specify the files which are to be executed by setting the proper check box. Use the "Select All" and "Deselect All" buttons, if necessary.
3. Define the order of the files in the list using the "Move Up" and "Move Down" buttons or dragging the required files. The scripts will be executed in this order.
4. Press "Okay" to apply changes and exit or "Cancel" to exit without applying the changes you have made.
5. 5.

Select Tools Devart Developer Tools Project Deploy.

#### Creating and executing SQL statements and scripts with SQL editor

To execute an SQL statement or script, first open a new SQL document by clicking on the "Create New SQL Editor" button on the toolbar. Type your query or script in it, and click the "Execute SQL" button. Query results and any error messages will be redirected to the common Output window. You can view the datasets returned from SELECT queries in the Data tab (select Data from the View menu if this tab is not yet visible). Note that dbForge Fusion for MySQL allows obtaining multiple result sets from SQL scripts.

#### Visually designing queries with the Query Builder

In an SQL document, you can switch to Design view to construct a query using Query Builder. In this mode you can create SELECT statements visually without using SQL. The Query Builder view is synchroni ed with the text view, and if you had a correct SELECT statement in the SQL editor, it is automatically inserted into the Query Builder. In the Query Builder you can drag and drop tables from the Database Explorer, use a special tabbed editor to setup JOIN statements as well as WHERE, GROUP BY, HAVING and ORDER BY clauses.

#### Opening and saving SQL documents

You can save your SQL document at any time for future use. SQL editor documents are saved with extension ".sql". Query Builder documents have extension ".qry". When opened, Query Builder table controls restore their original position on the data diagram.

#### Examining the SQL query execution plan

One of the most important factors to worry about when developing SQL queries is query performance. With dbForge Fusion for MySQL, you can easily evaluate and optimi e the performance of a critical query by inspecting it visually in Plan view. Just paste your query into an SQL document and switch to Plan view to see even the most complicated statements parsed by MySQL and presented in a tree with explanations of what every step in the plan does.

#### Viewing and editing data using grid based editor

The data of a table and view objects can be edited in a grid-based data editor. This data editor is accessible from object's popup menu or from the Data view. When you open the editor, it is automatically filled with the data contained in the object. Here you can edit data directly in a grid format. To insert a new row, press the Ins key. To delete a row, select it and press Del. Changes are stored until you commit them; to apply the changes made, press Enter, and to cancel all pending changes press Escape. To refresh data in a table choose "Refresh" from the popup menu.

#### Creating new components by dragging items from Database Explorer

You automatically create new components that reference existing resources by selecting a connection, table, view, stored procedure or package object in the Database Explorer and dragging it onto a form designer. Then the IDE will automatically create a new component that references the selected resource.

**Note:** Drag-n-drop support is not available for Delphi 2005.

#### Extended Integration Features with MyDAC Component Editors

dbForge Fusion for MySQL integrates with MyDAC to give you a number of extended design-time benefits.

- Drag-n-drop support for creating new components from some database objects
- Easy selection of existing connections in TMyConnection component
- Added "Find", "Debug", "Edit SQL", "Query Builder", and "Retrieve Data" verbs to component popup menus
- Standard SQL editor in all MyDAC components with SQL field properties is replaced with the full-



featured dbForge Fusion for MySQL SQL editor, complete with code completion, syntax highlighting, outlining, and other functionality.

**Complete Documentation**

dbForge Fusion for MySQL comes with comprehensive documentation that describes all aspects of using the software and contains a number of walkthroughs and reference topics.

There are several ways to open this documentation:

- Use the appropriate shortcut in Start menu, for instance, Start Programs Devart dbForge Fusion Documentation.
- Use command from the IDE menu: Tools Devart dbForge Fusion Help.
- Focus on any dbForge Fusion for MySQL window (for example, on the Database Explorer), and press F1.

## **16.22 MyBuilder Add-In**

To extend MyDAC design-time capabilities, MyBuilder Add-in is provided. It is an easy to use and versatile MyDAC design-time extension to manipulate data and database objects of MySQL. With MyBuilder Add-in you can build, execute, verify and optimize your SQL statements.

MyBuilder Add-in is embedded in IDE and can be called from its main menu, component editors and component popup menus.

Sometimes when you install or upgrade MyBuilder Add-in or upgrade MyDAC there is an error message during MyDAC design-time packages initialization. It says: 'Current version of MyBuilder Add-in is incompatible with MyDAC X.XX'. To solve this problem go to MyBuilder Add-in directory and view Requirements section in ReadMe.txt. There you will find the lowest MyDAC version compatible with the current add-in version. Now if your current MyDAC version number is lower than required by add-in, you should upgrade MyDAC and if current version is higher, then upgrade MyBuilder Add-in. In more rare cases you may need to upgrade both products.

### **See Also**

- [TMyBuilder](#)

## 16.23 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:

| T PE      | 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)

---

## **16.24 Database Specific Aspects of 64-bit Development**

### **MySQL Connectivity Aspects**

#### **Client mode:**

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 a MySQL 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 at design-time. To work with a MySQL database in run-time (64-bit application), you must have the 64-bit client library placed to the C:\Windows\System32 directory.

#### **DIRECT mode:**

Since there is no need to install client library for the DIRECT mode, the specifics of developing applications that use MyDAC as data access components, depends exclusively on peculiarities of each target platform.

## 17 Reference

This page shortly describes units that exist in MyDAC.

### 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="#">CRVio</a>                    | This unit contains classes, used for establishing HTTP connections.                 |
| <a href="#">DADump</a>                   | This unit contains the base class for the TMyDump component.                        |
| <a href="#">DALoader</a>                 | This unit contains the base class for the TMyLoader component.                      |
| <a href="#">DAScript</a>                 | This unit contains the base class for the TMyScript component.                      |
| <a href="#">DASQLMonitor</a>             | This unit contains the base class for the TMySQLMonitor 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.MyDac.DataAdapter</a> | This unit contains implementation of the MyDataAdapter class.                       |
| <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.                         |
| MemUtils                                 | This unit contains auxiliary procedures and functions used in the DAC code.         |
| <a href="#">MyAccess</a>                 | This unit contains implementation of most public classes of MyDAC.                  |
| <a href="#">MyBackup</a>                 | This unit contains implementation of the TMyBackup component.                       |
| <a href="#">MyBuilderClient</a>          | This unit contains implementation of the TMyBuilder class.                          |
| <a href="#">MyClasses</a>                | This unit contains implementation of the EMyError class.                            |
| <a href="#">MyConnectionPool</a>         | This unit contains the TMyConnectionPoolManager class for managing connection pool. |
| <a href="#">MyDacVcl</a>                 | This unit contains the visual constituent of MyDAC.                                 |
| <a href="#">MyDump</a>                   | This unit contains implementation of the TMyDump component.                         |
| <a href="#">MyEmbConnection</a>          | This unit contains implementation of the TMyEmbConnection component.                |



[MyLoader](#)

This unit contains implementation of the TMyLoader component.

[MyScript](#)

This unit contains implementation of the TMyScript component.

[MyServerControl](#)

This unit contains implementation of the TMyServerControl component.

[MySqlApi](#)

This unit contains implementation of the class.

[MySQLMonitor](#)

This unit contains implementation of the TMySQLMonitor component.

MySqlVio

This unit contains implementation of the TCRIOHandler class.

[VirtualTable](#)

This unit contains implementation of the TVirtualTable component.

## **17.1 CRAccess**

This unit contains base classes for accessing databases.

### **Classes**

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

### **Types**

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

### **Enumerations**

| <b>Name</b>                          | <b>Description</b>   |
|--------------------------------------|--|
| <a href="#">TCRIsoationLevel</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. |

---

## 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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.1.1.1 CRAccess.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

TObject

[TSharedObject](#)

**TCRCursor**

© 1997-2012 Devart. All Rights Reserved.

[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.  |

© 1997-2012 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. |

---

© 1997-2012 Devart. All Rights Reserved.

#### 17.1.2.1 CRAccess.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.

---

© 1997-2012 Devart. All Rights Reserved.

### 17.1.3 Enumerations

Enumerations in the **CRAccess** unit.

#### 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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.1.3.1 CRAccess.TCRIsolationLevel Enumeration

Specifies how to handle transactions containing database modifications.

#### Unit

[CRAccess](#)

#### Syntax

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

#### Values

| Value                  | Meaning   |
|------------------------|---|
| <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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.1.3.2 CRAccess.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. |

© 1997-2012 Devart. All Rights Reserved.

## **17.2 CRBatchMove**

This unit contains implementation of the TCRBatchMove component.

### **Classes**

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

### **Types**

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

### **Enumerations**

| <b>Name</b>                         | <b>Description</b>   |
|-------------------------------------|--|
| <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. |

---

## 17.2.1 Classes

Classes in the **CRBatchMove** unit.

### Classes

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

© 1997-2012 Devart. All Rights Reserved.

#### 17.2.1.1 CRBatchMove.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 MySQL Access page.

### Inheritance Hierarchy

TObject

**TCRBatchMove**

© 1997-2012 Devart. All Rights Reserved.

[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.   |

[KeyViolCount](#)

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

[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.

[MovedCount](#)

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

[ProblemCount](#)

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

[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.

**Methods****Name****Description**[Execute](#)

Performs the batch operation.

**Events****Name****Description**[OnBatchMoveProgress](#)

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

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TCRBatchMove** class.

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

**Public****Name****Description**[ChangedCount](#)

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

[KeyViolCount](#)

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

[MovedCount](#)

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

[ProblemCount](#)

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**



[AbortOnKeyViol](#)

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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
```

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

Performs the batch operation.

## Class

[TCRBatchMove](#)

## Syntax

**procedure** Execute;

## Remarks

Call the Execute method to perform the batch operation.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

---

## 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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.2.2.1 CRBatchMove.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.

© 1997-2012 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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.2.3.1 CRBatchMove.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.   |

© 1997-2012 Devart. All Rights Reserved.

#### 17.2.3.2 CRBatchMove.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.  |

© 1997-2012 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   |

## 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.     |
| <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   |

© 1997-2012 Devart. All Rights Reserved.

#### 17.3.1.1 CRDataTypeMap.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

TObject

[EDataTypeMappingError](#)

**EDataMappingError**

© 1997-2012 Devart. All Rights Reserved.

[EDataMappingError](#) class overview.

© 1997-2012 Devart. All Rights Reserved.

#### 17.3.1.2 CRDataTypeMap.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

**Inheritance Hierarchy**

TObject

**EDataTypeMappingError**

---

© 1997-2012 Devart. All Rights Reserved.

[EDataTypeMappingError](#) class overview.

---

© 1997-2012 Devart. All Rights Reserved.

## 17.3.1.3 CRDataTypeMap.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**

TObject

[EDataTypeMappingError](#)

**EInvalidDBTypeMapping**

---

© 1997-2012 Devart. All Rights Reserved.

[EInvalidDBTypeMapping](#) class overview.

---

© 1997-2012 Devart. All Rights Reserved.

## 17.3.1.4 CRDataTypeMap.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**

TObject

[EDataTypeMappingError](#)

**EInvalidFieldTypeMapping**

© 1997-2012 Devart. All Rights Reserved.

[EInvalidFieldTypeMapping](#) class overview.

© 1997-2012 Devart. All Rights Reserved.

#### 17.3.1.5 CRDataTypeMap.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

TObject

[EDataTypeMappingError](#)

**EUnsupportedDataTypeMapping**

© 1997-2012 Devart. All Rights Reserved.

[EUnsupportedDataTypeMapping](#) class overview.

© 1997-2012 Devart. All Rights Reserved.

#### 17.3.1.6 CRDataTypeMap.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);
```

#### Inheritance Hierarchy

TObject

**TMapRule**

© 1997-2012 Devart. All Rights Reserved.

[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 type                |
| <a href="#">FieldLength</a> | Delphi field length    |
| <a href="#">FieldName</a>   | field name in DataSet  |

[FieldScale](#)  
[IgnoreErrors](#)

Delphi field scale  
 Ignore data conversion errors.  
 Default value is False.

© 1997-2012 Devart. All Rights Reserved.

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 type   |
| <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.<br>Default value is False. |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Maximum DB field size

## Class

[TMapRule](#)

## Syntax

```
property DBLengthMax: Integer;
```

© 1997-2012 Devart. All Rights Reserved.

Minimum DB field size

## Class

[TMapRule](#)

## Syntax

```
property DBLengthMin: Integer;
```

© 1997-2012 Devart. All Rights Reserved.

Maximum DB field scale

## Class

[TMapRule](#)

## Syntax

```
property DBScaleMax: Integer;
```

© 1997-2012 Devart. All Rights Reserved.

Minimal DB field scale

**Class**

[TMapRule](#)

**Syntax**

```
property DBScaleMin: Integer;
```

---

© 1997-2012 Devart. All Rights Reserved.

DB type

**Class**

[TMapRule](#)

**Syntax**

```
property DBType: Word;
```

---

© 1997-2012 Devart. All Rights Reserved.

Delphi field length

**Class**

[TMapRule](#)

**Syntax**

```
property FieldLength: Integer;
```

---

© 1997-2012 Devart. All Rights Reserved.

field name in DataSet

**Class**

[TMapRule](#)

**Syntax**

```
property FieldName: string;
```

---

© 1997-2012 Devart. All Rights Reserved.

Delphi field scale

**Class**

[TMapRule](#)

**Syntax**

```
property FieldScale: Integer;
```

---

© 1997-2012 Devart. All Rights Reserved.

Ignore data conversion errors. Default value is False.

**Class**

[TMapRule](#)

**Syntax**

```
property IgnoreErrors: Boolean;
```

---



## **17.4 CREncryption**

This unit contains base classes for data encryption.

### **Classes**

| <b>Name</b>                  | <b>Description</b>   |
|------------------------------|--|
| <a href="#">TCREncryptor</a> | The class that performs data encryption and decryption in a client application using various <a href="#">encryption algorithms</a> . |

### **Enumerations**

| <b>Name</b>                            | <b>Description</b>  |
|--|---|
| <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.     |

---



## 17.4.1 Classes

Classes in the **CREncryption** unit.

### Classes

| Name                         | Description  |
|------------------------------|--|
| <a href="#">TCREncryptor</a> | The class that performs data encryption and decryption in a client application using various <a href="#">encryption algorithms</a> . |

© 1997-2012 Devart. All Rights Reserved.

#### 17.4.1.1 CREncryption.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) ;
```

### Inheritance Hierarchy

TObject

**TCREncryptor**

© 1997-2012 Devart. All Rights Reserved.

[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. |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TCREncryptor** class.

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

### Published

| Name | Description |
|------|-------------|
|------|-------------|

[DataHeader](#)

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

[EncryptionAlgorithm](#)

Specifies the algorithm of data encryption.

[HashAlgorithm](#)

Specifies the algorithm of generating hash data.

[InvalidHashAction](#)

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

[Password](#)

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

### See Also

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

© 1997-2012 Devart. All Rights Reserved.

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](#).

© 1997-2012 Devart. All Rights Reserved.

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](#).

© 1997-2012 Devart. All Rights Reserved.

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](#).

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

---

## 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.     |

© 1997-2012 Devart. All Rights Reserved.

#### 17.4.2.1 CREncryption.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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.4.2.2 CREncryption.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.                         |

© 1997-2012 Devart. All Rights Reserved.

#### 17.4.2.3 CREncryption.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. |

---

© 1997-2012 Devart. All Rights Reserved.

#### 17.4.2.4 CREncryption.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.                          |

---

© 1997-2012 Devart. All Rights Reserved.

## **17.5 CRVio**

This unit contains classes, used for establishing HTTP connections.

### **Classes**

| <b>Name</b>                   | <b>Description</b>   |
|-------------------------------|--|
| <a href="#">THttpOptions</a>  | The class contains settings for HTTP connection.   |
| <a href="#">TProxyOptions</a> | This class is used when connecting through proxy server to establish an HTTP connection. |

## 17.5.1 Classes

Classes in the **CRVio** unit.

### Classes

| Name                          | Description  |
|-------------------------------|--|
| <a href="#">THttpOptions</a>  | The class contains settings for HTTP connection.   |
| <a href="#">TProxyOptions</a> | This class is used when connecting through proxy server to establish an HTTP connection. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.5.1.1 CRVio.THttpOptions Class

The class contains settings for HTTP connection.

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

### Unit

[CRVio](#)

### Syntax

```
THttpOptions = class (TPersistent);
```

### Remarks

The THttpOptions class contains settings for HTTP connection.

For more information on HTTP tunneling refer to the [Network Tunneling](#) article.

### Inheritance Hierarchy

```
TObject
  THttpOptions
```

### See Also

- [Network Tunneling](#)

© 1997-2012 Devart. All Rights Reserved.

[THttpOptions](#) class overview.

### Properties

| Name                         | Description   |
|------------------------------|---|
| <a href="#">Password</a>     | Holds the password for HTTP authentication.                               |
| <a href="#">ProxyOptions</a> | Holds a TProxyOptions object that contains settings for proxy connection. |
| <a href="#">Url</a>          | Holds the url of the tunneling PHP script.                                |
| <a href="#">Username</a>     | Holds the user name for HTTP authentication.                              |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **THttpOptions** class.

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

### Published



| Name                         | Description   |
|------------------------------|---|
| <a href="#">Password</a>     | Holds the password for HTTP authori ation.                                |
| <a href="#">ProxyOptions</a> | Holds a TProxyOptions object that contains settings for proxy connection. |
| <a href="#">Url</a>          | Holds the url of the tunneling PHP script.                                |
| <a href="#">Username</a>     | Holds the user name for HTTP authori ation.                               |

### See Also

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

---

© 1997-2012 Devart. All Rights Reserved.

Holds the password for HTTP authori ation.

### Class

[THttpOptions](#)

### Syntax

```
property Password: string;
```

### Remarks

The Password property holds the password for HTTP authori ation.

---

© 1997-2012 Devart. All Rights Reserved.

Holds a TProxyOptions object that contains settings for proxy connection.

### Class

[THttpOptions](#)

### Syntax

```
property ProxyOptions: TProxyOptions;
```

### Remarks

The ProxyOptions property holds a TProxyOptions object that contains settings for proxy connection. If it is necessary to connect to server in another network, sometimes the client can reach it only through proxy. In this case in addition to connection string you have to setup ProxyOptions.

---

© 1997-2012 Devart. All Rights Reserved.

Holds the url of the tunneling PHP script.

### Class

[THttpOptions](#)

### Syntax

```
property Url: string;
```

### Remarks

The Url property holds the url of the tunneling PHP script. For example, if the script is in the server root, the url can be the following: http://server/tunnel.php.

---

© 1997-2012 Devart. All Rights Reserved.

Holds the user name for HTTP authentication.

### Class

[THttpOptions](#)

### Syntax

```
property Username: string;
```

### Remarks

The Username property holds the user name for HTTP authentication.

© 1997-2012 Devart. All Rights Reserved.

#### 17.5.1.2 CRVio.TProxyOptions Class

This class is used when connecting through proxy server to establish an HTTP connection. For a list of all members of this type, see [TProxyOptions](#) members.

### Unit

[CRVio](#)

### Syntax

```
TProxyOptions = class(TPersistent);
```

### Remarks

The TProxyOptions class is used when connecting through proxy server to establish an HTTP connection.

### Inheritance Hierarchy

TObject  
**TProxyOptions**

© 1997-2012 Devart. All Rights Reserved.

[TProxyOptions](#) class overview.

### Properties

| Name                     | Description  |
|--------------------------|--|
| <a href="#">Hostname</a> | Holds the host name or IP address to connect to proxy server.            |
| <a href="#">Password</a> | Holds the password for the proxy server account.                         |
| <a href="#">Port</a>     | Used to specify the port number for TCP/IP connection with proxy server. |
| <a href="#">Username</a> | Holds the proxy server account name.                                     |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TProxyOptions** class.

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

### Published

| Name                     | Description  |
|--------------------------|--|
| <a href="#">Hostname</a> | Holds the host name or IP address to connect to proxy server.            |
| <a href="#">Password</a> | Holds the password for the proxy server account.                         |
| <a href="#">Port</a>     | Used to specify the port number for TCP/IP connection with proxy server. |

[Username](#)

Holds the proxy server account name.

### See Also

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

---

© 1997-2012 Devart. All Rights Reserved.

Holds the host name or IP address to connect to proxy server.

### Class

[TProxyOptions](#)

### Syntax

```
property Hostname: string;
```

### Remarks

The Hostname property holds the host name or IP address to connect to proxy server.

---

© 1997-2012 Devart. All Rights Reserved.

Holds the password for the proxy server account.

### Class

[TProxyOptions](#)

### Syntax

```
property Password: string;
```

### Remarks

The Password property holds the password for the proxy server account.

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify the port number for TCP/IP connection with proxy server.

### Class

[TProxyOptions](#)

### Syntax

```
property Port: integer default 0;
```

### Remarks

Use the Port property to specify the port number for TCP/IP connection with proxy server.

---

© 1997-2012 Devart. All Rights Reserved.

Holds the proxy server account name.

### Class

[TProxyOptions](#)

### Syntax

```
property Username: string;
```

### Remarks

The Username property holds the proxy server account name.

---



## 17.6 DADump

This unit contains the base class for the TMyDump 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  |
|---|--|
| <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. |

## 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.                                       |

© 1997-2012 Devart. All Rights Reserved.

#### 17.6.1.1 DADump.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.

### Inheritance Hierarchy

TObject  
**TDADump**

© 1997-2012 Devart. All Rights Reserved.

[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. |

[BackupQuery](#)

Dumps the results of a particular query.

[BackupToFile](#)

Dumps database objects to the specified file.

[BackupToStream](#)

Dumps database objects to the stream.

[Restore](#)

Executes a script contained in the SQL property.

[RestoreFromFile](#)

Executes a script from a file.

[RestoreFromStream](#)

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. |
| <a href="#">OnError</a>           | Occurs when MySQL 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.                      |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TDADump** class.

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

## Public

| Name                       | Description   |
|----------------------------|---|
| <a href="#">Connection</a> | Used to specify a connection object that will be used to connect to a data store. |
| <a href="#">Options</a>    | Used to specify the behaviour of a TDADump component.                             |

## Published

| Name                       | Description  |
|----------------------------|--|
| <a href="#">Debug</a>      | Used to display executing statement, all its parameters' values, and the type of parameters. |
| <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.   |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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. |

©

1997-2012 Devart. All Rights Reserved.



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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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 MySQL 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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

Occurs when MySQL raises some error on [Restore](#).

### Class

[TDADump](#)

### Syntax

**property** OnError: [TOnErrorEvent](#);

### Remarks

The OnError event occurs when MySQL 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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

### 17.6.1.2 DADump.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);

## Inheritance Hierarchy

TObject

**TDADumpOptions**

© 1997-2012 Devart. All Rights Reserved.

[TDADumpOptions](#) class overview.

## Properties

| 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. |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TDADumpOptions** class.

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

## Published

| 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.

### See Also

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

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.6.2.1 DADump.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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.6.2.2 DADump.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.

© 1997-2012 Devart. All Rights Reserved.



## 17.7 DALoader

This unit contains the base class for the TMyLoader 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.      |

## 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.   |

© 1997-2012 Devart. All Rights Reserved.

#### 17.7.1.1 DALoader.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.

### Inheritance Hierarchy

```

TObject
  TDAColumn
  
```

### See Also

- [TDALoader](#)
- [TDAColumns](#)

© 1997-2012 Devart. All Rights Reserved.

[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.         |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TDAColumn** class.

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

### Published

| Name | Description |
|------|-------------|
|------|-------------|

[FieldType](#)

Used to specify the types of values that will be loaded.

[Name](#)

Used to specify the field name of loading table.

### See Also

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

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

#### 17.7.1.2 DALoader.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.

### Inheritance Hierarchy

TObject  
**TDAColumns**

## See Also

- [TDALoader](#)
- [TDAColumn](#)

© 1997-2012 Devart. All Rights Reserved.

[TDAColumns](#) class overview.

## Properties

| Name                  | Description                        |
|-----------------------|------------------------------------|
| <a href="#">Items</a> | Used to access individual columns. |

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

### 17.7.1.3 DALoader.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.

## Inheritance Hierarchy

TObject  
    **TDALoader**

## See Also

- [TMyLoader](#)

© 1997-2012 Devart. All Rights Reserved.

[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   |
|---------------------------------|---|
| <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.   |

## 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.LoadFromDataSet</a> method is needed. |
| <a href="#">OnPutData</a>       | Occurs when putting loading data by rows is needed.   |

© 1997-2012 Devart. All Rights Reserved.

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 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.                |

### See Also

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

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- M:Devart.Dac.TCustomDAConnection.GetTableNames(Borland.Vcl.TStrings,System.Boolean)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.  |

© 1997-2012 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](#)

© 1997-2012 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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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. [TMyLoader](#) will flush data to MySQL when it is needed (see [TMyLoader.RowsPerQuery](#)).

To start loading, call the [Load](#) method.

## Example

This handler loads 1000 rows.

```
procedure TfmMain.PutData(Sender: TDALoader);  
var  
    Count: Integer;  
    i: Integer;  
begin  
    Count := StrToInt(edRows.Text);  
    for i := 1 to Count do begin  
        Sender.PutColumnData(0, i, 1);  
        Sender.PutColumnData(1, i, Random(100));  
        Sender.PutColumnData(2, i, Random*100);  
        Sender.PutColumnData(3, i, 'abc01234567890123456789');  
        Sender.PutColumnData(4, i, Date);  
    end;  
end;
```

## See Also

- [TDALoader.PutColumnData](#)
- [Load](#)
- [OnGetColumnData](#)

---

© 1997-2012 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.      |

© 1997-2012 Devart. All Rights Reserved.

#### 17.7.2.1 DALoader.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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.7.2.2 DALoader.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.

© 1997-2012 Devart. All Rights Reserved.

## 17.7.2.3 DALoader.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.

## 17.8 DAScript

This unit contains the base class for the TMyScript 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.  |
| <a href="#">TBeforeStatementExecuteEvent</a> | This type is used for the <a href="#">TDAScript.BeforeExecute</a> event. |
| <a href="#">TOnErrorEvent</a>                | This type is used for the <a href="#">TDAScript.OnError</a> event.       |

### Enumerations

| Name                         | Description  |
|------------------------------|--|
| <a href="#">TErrorAction</a> | Indicates the action to take when the OnError handler exits. |

---

## 17.8.1 Classes

Classes in the **DAScript** unit.

### 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.                            |

© 1997-2012 Devart. All Rights Reserved.

#### 17.8.1.1 DAScript.TDAScript Class

Makes it possible to execute several SQL statements one by one.  
For a list of all members of this type, see [TDAScript](#) members.

### Unit

[DAScript](#)

### Syntax

```
TDAScript = 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 TDAScript 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, 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.

### Inheritance Hierarchy

TObject  
    **TDAScript**

### See Also

- [TCustomDASQL](#)

© 1997-2012 Devart. All Rights Reserved.

[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.   |
| <a href="#">StartPos</a>    | Used to get the start position of the current statement in a script. |
| <a href="#">Statements</a>  | 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 MySQL raises an error.   |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TDAScript** class.

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

## Public

| 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="#">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="#">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.   |
| <a href="#">StartPos</a>    | Used to get the start position of the current statement in a script. |
| <a href="#">Statements</a>  | 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](#)

© 1997-2012 Devart. All Rights Reserved.

Used to specify the connection in which the script will be executed.

## Class

[TDAScript](#)

## 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](#)

© 1997-2012 Devart. All Rights Reserved.

Refers to a dataset that holds the result set of query execution.

## Class

[TDAScript](#)

## 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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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 if the Delimiter property's default value is semicolon. Note that slash must be the first character in line.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

Used to get the offset in the last line of the current statement.

## Class

[TDA Script](#)

## Syntax

```
property EndOffset: Int64;
```

## Remarks

Use the EndOffset property to get the offset in the last line of the current statement.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to get or set script text.

### Class

[TDAScript](#)

### Syntax

```
property SQL: TStrings;
```

### Remarks

Use the SQL property to get or set script text.

---

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

Used to get the offset in the first line of the current statement.

### Class

[TDA Script](#)

### Syntax

**property** StartOffset: Int64;

### Remarks

Use the StartOffset property to get the offset in the first line of the current statement.

© 1997-2012 Devart. All Rights Reserved.

Used to get the start position of the current statement in a script.

### Class

[TDA Script](#)

### 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.

© 1997-2012 Devart. All Rights Reserved.

Contains a list of statements obtained from the SQL property.

### Class

[TDA Script](#)

### Syntax

**property** Statements: [TDA Statements](#);

### 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;
    end;
end;
```

## See Also

- [TDASentences](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

Stops script execution.

## Class

[TDAScript](#)

## Syntax

```
procedure BreakExec; virtual;
```

## Remarks

Call the BreakExec method to stop script execution.

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Executes a script.

### Class

[TDA Script](#)

### Syntax

```
procedure Execute; virtual;
```

### Remarks

Call the Execute method to execute a script. If MySQL raises an error, the OnError event occurs.

### See Also

- [ExecuteNext](#)
  - [OnError](#)
  - [ErrorOffset](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Executes SQL statements contained in a file.

### Class

[TDA Script](#)

### 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.

---

© 1997-2012 Devart. All Rights Reserved.

Executes the next statement in the script and then stops.

## Class

[TDA Script](#)

## 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 MySQL raises an error, the OnError event occurs.

## See Also

- [Execute](#)
- [OnError](#)
- [ErrorOffset](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Finds a Macro with the name passed in Name.

## Class

[TDAScript](#)

## 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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Events of the **TDAScript** class.

For a complete list of the **TDAScript** class members, see the [TDAScript 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 MySQL raises an error.   |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.



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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

Occurs when MySQL raises an error.

### Class

[TDA Script](#)

### Syntax

**property** OnError: [TOnErrorEvent](#);

### Remarks

Occurs when MySQL 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](#)

---

© 1997-2012 Devart. All Rights Reserved.

#### 17.8.1.2 DAscript.TDAStatement 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 [TDAStatement](#) members.

### Unit

[DAScript](#)

### Syntax

```
TDASstatement = class (TCollectionItem);
```

## Remarks

TDAScript contains SQL statements, represented as TDASstatement objects. The TDASstatement class has attributes and methods for controlling single SQL statement of a script.

## Inheritance Hierarchy

```
TObject
  TDASstatement
```

## See Also

- [TDAScript](#)
- [TDASstatements](#)

© 1997-2012 Devart. All Rights Reserved.

[TDASstatement](#) 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 TDAScript 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.          |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TDASstatement** class.

For a complete list of the **TDASstatement** class members, see the [TDASstatement 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.                              |

[Params](#)

Contains parameters for an SQL statement.

[Script](#)

Used to determine the TDAScript 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

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

---

© 1997-2012 Devart. All Rights Reserved.

Used to determine the number of the last statement line in a script.

### Class

[TDASStatement](#)

### Syntax

```
property EndLine: integer;
```

### Remarks

Use the EndLine property to determine the number of the last statement line in a script.

---

© 1997-2012 Devart. All Rights Reserved.

Used to get the offset in the last line of the statement.

### Class

[TDASStatement](#)

### Syntax

```
property EndOffset: integer;
```

### Remarks

Use the EndOffset property to get the offset in the last line of the statement.

---

© 1997-2012 Devart. All Rights Reserved.

Used to get the end position of the statement in a script.

### Class

[TDASStatement](#)

### 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.

---

© 1997-2012 Devart. All Rights Reserved.

Used to avoid execution of a statement.

**Class**

[TDAStatement](#)

**Syntax**

**property** Omit: boolean;

**Remarks**

Set the Omit property to True to avoid execution of a statement.

---

© 1997-2012 Devart. All Rights Reserved.

Contains parameters for an SQL statement.

**Class**

[TDAStatement](#)

**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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to determine the TDA Script object the SQL Statement belongs to.

**Class**

[TDAStatement](#)

**Syntax**

**property** Script: [TDA Script](#);

**Remarks**

Use the Script property to determine the TDA Script object the SQL Statement belongs to.

---

© 1997-2012 Devart. All Rights Reserved.

Used to get or set the text of an SQL statement.

**Class**

[TDAStatement](#)

**Syntax**

**property** SQL: string;

**Remarks**

Use the SQL property to get or set the text of an SQL statement.

---

© 1997-2012 Devart. All Rights Reserved.

Used to determine the number of the first statement line in a script.

## Class

[TDASentence](#)

## Syntax

```
property StartLine: integer;
```

## Remarks

Use the StartLine property to determine the number of the first statement line in a script.

© 1997-2012 Devart. All Rights Reserved.

Used to get the offset in the first line of a statement.

## Class

[TDASentence](#)

## Syntax

```
property StartOffset: integer;
```

## Remarks

Use the StartOffset property to get the offset in the first line of a statement.

© 1997-2012 Devart. All Rights Reserved.

Used to get the start position of the statement in a script.

## Class

[TDASentence](#)

## 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.

© 1997-2012 Devart. All Rights Reserved.

### 17.8.1.3 DAScript.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 statements in its Items array. The Count property contains the number of statements in the collection. Use TDASentences class to manipulate script SQL statements.

## Inheritance Hierarchy

TObject

**TDASentences**

## See Also

- [TDAScript](#)
- [TDAStatement](#)

© 1997-2012 Devart. All Rights Reserved.

[TDAStatements](#) class overview.

## Properties

| Name                  | Description                                |
|-----------------------|--|
| <a href="#">Items</a> | Used to access separate script statements. |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TDAStatements** class.

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

## Public

| Name                  | Description                                |
|-----------------------|--|
| <a href="#">Items</a> | Used to access separate script statements. |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Used to access separate script statements.

## Class

[TDAStatements](#)

## Syntax

```
property Items[Index: Integer]: TDAStatement; 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 [TDAStatement](#).

## See Also

- [TDAStatement](#)

© 1997-2012 Devart. All Rights Reserved.

## 17.8.2 Types

Types in the **DAScript** unit.

### Types

| Name   | Description  |
|--|--|
| <a href="#">TAfterStatementExecuteEvent</a>  | This type is used for the <a href="#">TDAScript.AfterExecute</a> event.  |
| <a href="#">TBeforeStatementExecuteEvent</a> | This type is used for the <a href="#">TDAScript.BeforeExecute</a> event. |
| <a href="#">TOnErrorEvent</a>                | This type is used for the <a href="#">TDAScript.OnError</a> event.       |

© 1997-2012 Devart. All Rights Reserved.

#### 17.8.2.1 DAScript.TAfterStatementExecuteEvent Procedure Reference

This type is used for the [TDAScript.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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.8.2.2 DAScript.TBeforeStatementExecuteEvent Procedure Reference

This type is used for the [TDAScript.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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.8.2.3 DAScript.TOnErrorEvent Procedure Reference

This type is used for the [TDAScript.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.

---



### 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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.8.3.1 DAScript.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.<br>HandleException method. |
| <b>eaFail</b>      | Abort execution and display an error message.  |

© 1997-2012 Devart. All Rights Reserved.

## 17.9 DASQLMonitor

This unit contains the base class for the TMySQLMonitor 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 displayed.  |

## 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.  |

© 1997-2012 Devart. All Rights Reserved.

#### 17.9.1.1 DASQLMonitor.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.

### Inheritance Hierarchy

TObject  
    **TCustomDASQLMonitor**

© 1997-2012 Devart. All Rights Reserved.

[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. |

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

Used to set options for dbMonitor.

## Class

[TCustomDASQLMonitor](#)

## Syntax

```
property DBMonitorOptions: TDBMonitorOptions;
```

## Remarks

Use DBMonitorOptions to set options for dbMonitor.

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

#### 17.9.1.2 DASQLMonitor.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);
```

### Inheritance Hierarchy

TObject

**TDBMonitorOptions**

---

© 1997-2012 Devart. All Rights Reserved.

[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.   |

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.9.2.1 DASQLMonitor.TDATraceFlags Set

Represents the set of [TDATraceFlag](#).

### Unit

[DASQLMonitor](#)

### Syntax

```
TDATraceFlags = set of TDATraceFlag;
```

© 1997-2012 Devart. All Rights Reserved.

#### 17.9.2.2 DASQLMonitor.TMonitorOptions Set

Represents the set of [TMonitorOption](#).

### Unit

[DASQLMonitor](#)

### Syntax

```
TMonitorOptions = set of TMonitorOption;
```

© 1997-2012 Devart. All Rights Reserved.

#### 17.9.2.3 DASQLMonitor.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.

© 1997-2012 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.  |

© 1997-2012 Devart. All Rights Reserved.

#### 17.9.3.1 DASQLMonitor.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>       | If this flag is set, then just before sending a query to the server, OnSQL event is called additionally. The difference from usual call is that the query is already completely decoded, i.e. parameters are quoted and included into the text of the query. If to use MySQL 4.1 protocol with preparing, a value of this flag will be ignored. |
| <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.  |

© 1997-2012 Devart. All Rights Reserved.

#### 17.9.3.2 DASQLMonitor.TMonitorOption Enumeration

Used to define where information from SQLMonitor will be dispalyed.

#### 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 href="#">DBMonitor</a> .   |
| <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.  |

---

## 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="#">TCustomDAConnection</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.   |
| <a href="#">TDAMapRule</a>            | Class that forms rules for Data Type Mapping.  |
| <a href="#">TDAMapRules</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="#">TDAMetaData</a>           | A class for retrieving metainformation of the specified database objects in the form of dataset.                               |
| <a href="#">TDAParam</a>              | A class that forms objects to represent the values of the <a href="#">parameters set</a> .                                     |
| <a href="#">TDAParams</a>             | This class is used to manage a list of TDAParam objects for an object that uses field parameters.                              |
| <a href="#">TDATransaction</a>        | A base class that implements functionality for controlling transactions.   |
| <a href="#">TMacro</a>                | Object that represents the value of a macro.   |
| <a href="#">TMacros</a>               | Controls a list of TMacro objects for the <a href="#">TCustomDASQL.Macros</a> or <a href="#">TCustomDADataset</a> 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.  |
| <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> .   |
| <a href="#">TUpdateExecuteEvent</a>      | This type is used for the <a href="#">TCustomDADataSet.AfterUpdateExecute</a> and <a href="#">TCustomDADataSet.BeforeUpdateExecute</a> events. |

## Enumerations

| Name                           | Description   |
|--------------------------------|---|
| <a href="#">TLabelSet</a>      | Sets the language of labels in the connect dialog.            |
| <a href="#">TLockMode</a>      | This enumeration defines a type of an editing record locking. |
| <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="#">BaseSQLOldBehavior</a> | After assigning SQL text and modifying it by <a href="#">AddWhere</a> , <a href="#">DeleteWhere</a> , and <a href="#">SetOrderBy</a> , all subsequent changes of the SQL property will not be reflected in the BaseSQL property. |
| <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.  |

[SQLGeneratorCompatibility](#)

The value of the [TCustomDADataSet.BaseSQL](#) property is used to complete the refresh SQL statement, if the manually assigned [TCustomDAUpdateSQL.RefreshSQL](#) property contains only WHERE clause.

### 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="#">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="#">TCustomDAConnection</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.   |
| <a href="#">TDAMapRule</a>            | Class that forms rules for Data Type Mapping.  |
| <a href="#">TDAMapRules</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="#">TDAMetaData</a>           | A class for retrieving metainformation of the specified database objects in the form of dataset.                               |
| <a href="#">TDAParam</a>              | A class that forms objects to represent the values of the <a href="#">parameters set</a> .                                     |
| <a href="#">TDAParams</a>             | This class is used to manage a list of TDAParam objects for an object that uses field parameters.                              |
| <a href="#">TDATransaction</a>        | A base class that implements functionality for controlling transactions.   |
| <a href="#">TMacro</a>                | Object that represents the value of a macro.   |
| <a href="#">TMacros</a>               | Controls a list of TMacro objects for the <a href="#">TCustomDASQL.Macros</a> or <a href="#">TCustomDADataset</a> components.  |
| <a href="#">TPoolingOptions</a>       | This class allows setting up the behaviour of the connection pool.   |

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.1.1 DBAccess.EDAEError 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 [EDAEError](#) members.

#### Unit

[DBAccess](#)

#### Syntax

```
EDAEError = class (EDatabaseError) ;
```

#### Remarks

EDAEError is a base class for exceptions that are raised when an error occurs on the server side.

#### Inheritance Hierarchy

TObject  
**EDAEError**

© 1997-2012 Devart. All Rights Reserved.

[EDAEError](#) 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. |

© 1997-2012 Devart. All Rights Reserved.

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

#### Public

| 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. |

#### See Also

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

© 1997-2012 Devart. All Rights Reserved.

Contains the component that caused the error.

#### Class

[EDAEError](#)

#### Syntax

```
property Component: TObject;
```

#### Remarks

The Component property contains the component that caused the error.

© 1997-2012 Devart. All Rights Reserved.

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 MySQL. This value is always positive.

See MySQL server Reference Manual.

## See Also

- [EMyError](#)

© 1997-2012 Devart. All Rights Reserved.

### 17.10.1.2 DBAccess.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.

## Inheritance Hierarchy

TObject

**TCRDataSource**

© 1997-2012 Devart. All Rights Reserved.

[TCRDataSource](#) class overview.

© 1997-2012 Devart. All Rights Reserved.

### 17.10.1.3 DBAccess.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.

## Inheritance Hierarchy

TObject  
    **TCustomConnectDialog**

© 1997-2012 Devart. All Rights Reserved.

[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.  |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TCustomConnectDialog** class.

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

## Public

| 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.   |

### See Also

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

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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 ConnectDialog: 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 MyDAC installation directory.

### See Also

- [GetServerList](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.1.4 DBAccess.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.

### Inheritance Hierarchy

TObject

**TCustomDAConnection**

© 1997-2012 Devart. All Rights Reserved.

[TCustomDAConnection](#) class overview.

### Properties

| Name                           | Description   |
|--------------------------------|---|
| <a href="#">ConnectDialog</a>  | Allows to link a <a href="#">TCustomConnectDialog</a> component.                      |
| <a href="#">ConvertEOL</a>     | Allows customizing 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="#">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. |

© 1997-2012 Devart. All Rights Reserved.

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.                      |
| <a href="#">ConvertEOL</a>    | Allows customizing 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.  |

[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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

Allows customi ing 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 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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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 MyDacVcl unit appears to the uses clause.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

©

1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#), [ValidateConnectionLifeTime](#), [Server](#), [Username](#), [Password](#), [TCustomMyConnection.Database](#), [TCustomMyConnection.IsolationLevel](#), [TMyConnection.Port](#), [TMyConnection.IOHandler](#), [TCustomMyConnection.ConnectionTimeout](#), [TMyConnectionOptions.Compress](#), [TMyConnectionOptions.Direct](#), [TMyConnectionOptions.Embedded](#), [TMyConnectionOptions.Protocol](#), [TCustomMyConnectionOptions.Charset](#), [TCustomMyConnectionOptions.UseUnicode](#), [TCustomMyConnectionOptions.NumericType](#).

**Note:** Using Pooling := True can cause errors with working with temporary tables.

## See Also

- [Username](#)
- [Password](#)
- [PoolingOptions](#)
- [Connection Pooling](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

©

1997-2012 Devart. All Rights Reserved.

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. If this property is not set, MyDAC tries to connect to Localhost.  
The Server property can be used only if [TMyConnectionOptions.Embedded](#) is set to False.

## See Also

- [Username](#)

- [Password](#)
- [TMyConnection.Port](#)

© 1997-2012 Devart. All Rights Reserved.

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, MyDAC tries to connect with the empty 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](#)

© 1997-2012 Devart. All Rights Reserved.

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="#">MonitorMessage</a>     | Sends a specified message through the <a href="#">TCustomDASQLMonitor</a> component. |

[RemoveFromPool](#)

Marks the connection that should not be returned to the pool after disconnect.

[Rollback](#)

Discards all current data changes and ends transaction.

[StartTransaction](#)

Begins a new user transaction.

### See Also

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

© 1997-2012 Devart. All Rights Reserved.

Applies changes in datasets.

### Class

[TCustomDAConnection](#)

### Overload List

| Name  | Description                                  |
|---|--|
| <a href="#">ApplyUpdates</a>                                      | Applies changes from all active datasets.    |
| <a href="#">ApplyUpdates(DataSets: array of TCustomDADataSet)</a> | Applies changes from the specified datasets. |

© 1997-2012 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](#)

© 1997-2012 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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
  - [TCustomMyDataSet.FetchAll](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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 rolls back 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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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:= MyMyConnection1.ExecProc('My Sum', [10, 20]);  
Label2.Caption:= MyMyConnection1.ParamByName('Result').AsString;
```

## See Also

- [ExecProcEx](#)
- [ExecSQL](#)



- [ExecSQLEx](#)

© 1997-2012 Devart. All Rights Reserved.

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 provide values only for the first and fourth parameters, you should call ExecProcEx in the following way:

```
Connection.ExecProcEx('Some_Stored_Procedure', ['Param_Name1', 'Param_Valu
```

## See Also

- [ExecSQL](#)
- [ExecSQLEx](#)
- [ExecProc](#)

© 1997-2012 Devart. All Rights Reserved.

Executes a SQL statement with parameters.

## Class

[TCustomDAConnection](#)

## Syntax

```
function ExecSQL(Text: string): variant; overload; function ExecSQL
(Text: string; const Params: array of variant): variant;
overload; 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 function having data type dtString.  
Otherwise returns Null.

**Remarks**

Use the ExecSQL method to execute any SQL statement outside the [TCustomDADataset](#) or [TCustomDASQL](#) components. Supply the Params array with the values of parameters arranged in the same order as they appear in a SQL statement which itself is passed to the Text string parameter.

**Note:** If a query doesn't have parameters (Params.Count = 0), this query will be executed faster.

**See Also**

- [ExecSQLEx](#)
- [ExecProc](#)
- [TCustomMyConnection.ExecSQL](#)

© 1997-2012 Devart. All Rights Reserved.

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

```
MyConnection.ExecSQLEx('begin :A:= :B + :C; end;',  
    ['A', 0, 'B', 5, 'C', 3]);  
A:= MyConnection.ParamByName('A').AsInteger;
```

### See Also

- [ExecSQL](#)

---

© 1997-2012 Devart. All Rights Reserved.

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

- M:Devart.Dac.TCustomDAConnection.GetTableNames(Borland.Vcl.TStrings,System.Boolean)
- [GetStoredProcNames](#)
- [TCustomMyConnection.Database](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
  - M:Devart.Dac.TCustomDAConnection.GetTableNames(Borland.Vcl.TStrings,System.Boolean)
  - M:Devart.Dac.TCustomDAConnection.GetTableNames(Borland.Vcl.TStrings,System.Boolean)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- [TCustomMyDataSet.FetchAll](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- [TCustomMyConnection.IsolationLevel](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.1.5 DBAccess.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

TObject

[TMemDataSet](#)

**TCustomDADataset**

© 1997-2012 Devart. All Rights Reserved.

[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.  |
| <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="#">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="#">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="#">RowsAffected</a>  | Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.  |
| <a href="#">SQL</a>   | Used to provide a SQL statement that a query component executes when its Open method is called.  |
| <a href="#">SQLDelete</a>   | 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.   |

[LocateEx](#) (inherited from [TMemDataSet](#))

[Lock](#)

[MacroByName](#)

[ParamByName](#)

[Prepare](#)

[RefreshRecord](#)

[RestoreSQL](#)

[RestoreUpdates](#) (inherited from [TMemDataSet](#))

[Resync](#)

[RevertRecord](#) (inherited from [TMemDataSet](#))

[SaveSQL](#)

[SaveToXML](#) (inherited from [TMemDataSet](#))

[SetOrderBy](#)

[SQLSaved](#)

[UnLock](#)

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Overloaded. Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet.

Locks the current record.

Finds a Macro with the name passed in Name.

Sets or uses parameter information for a specific parameter based on its name.

Allocates, opens, and parses cursor for a query.

Actuali es field values for the current record.

Restores the SQL property modified by AddWhere and SetOrderBy.

Marks all records in the cache of updates as unapplied.

Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.

Cancels changes made to the current record when cached updates are enabled.

Saves the SQL property value to BaseSQL.

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Builds an ORDER BY clause of a SELECT statement.

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

Releases a record lock.

Frees the resources allocated for a previously prepared query on the server and client sides.

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.

## 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.                     |

© 1997-2012 Devart. All Rights Reserved.

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.                |

[IndexFieldNames](#) (inherited from [TMemDataSet](#))

[IsQuery](#)

[KeyFields](#)

[LocalConstraints](#) (inherited from [TMemDataSet](#))

[LocalUpdate](#) (inherited from [TMemDataSet](#))

[Locate](#) (inherited from [TMemDataSet](#))

[LocateEx](#) (inherited from [TMemDataSet](#))

[MacroCount](#)

[Macros](#)

[MasterFields](#)

[MasterSource](#)

[OnUpdateError](#) (inherited from [TMemDataSet](#))

[OnUpdateRecord](#) (inherited from [TMemDataSet](#))

[Options](#)

[ParamCheck](#)

[ParamCount](#)

[Params](#)

[Prepare](#) (inherited from [TMemDataSet](#))

[Prepared](#) (inherited from [TMemDataSet](#))

[ReadOnly](#)

Used to get or set the list of fields on which the recordset is sorted.

Used to check whether SQL statement returns rows.

Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.

Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

Used to prevent implicit update of rows on database server.

Overloaded. Searches a dataset for a specific record and positions the cursor on it.

Overloaded. Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet.

Used to get the number of macros associated with the Macros property.

Makes it possible to change SQL queries easily.

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.

Used to specify the data source component which binds current dataset to the master one.

Occurs when an exception is generated while cached updates are applied to a database.

Occurs when a single update component can not handle the updates.

Used to specify the behaviour of TCustomDADataset object.

Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.

Used to indicate how many parameters are there in the Params property.

Used to view and set parameter names, values, and data types dynamically.

Allocates resources and creates field components for a dataset.

Determines whether a query is prepared for execution or not.

Used to prevent users from updating, inserting, or deleting data in the dataset.

[RefreshOptions](#)

Used to indicate when the editing record is refreshed.

[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.

[RowsAffected](#)

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

[SaveToXML](#) (inherited from [TMemDataSet](#))

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](#)

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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 MyDacVcl 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 MyDacVcl (MyDacClx under Kylix) unit to your project.

### See Also

- [TCustomDASQL.Debug](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

Used to change the WHERE clause of SELECT statement and reopen a query.

### Class

[TCustomDADataset](#)

### 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](#)

---

© 1997-2012 Devart. All Rights Reserved.

Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.

### Class

[TCustomDADataset](#)

### 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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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 uses the metainformation sent by the server together with data.

### See Also

- [SQLDelete](#)
- [SQLInsert](#)
- [SQLRefresh](#)
- [SQLUpdate](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

Makes it possible to change SQL queries easily.

### Class

[TCustomDADataset](#)

### Syntax

**property** Macros: [TMacros](#) stored False;

## Remarks

With the help of macros you can easily change SQL query text at design- or runtime. Marcos 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

```
MyQuery.SQL:= 'SELECT * FROM Dept ORDER BY &Order';  
MyQuery.MacroByName('Order').Value:= 'DeptNo';  
MyQuery.Open;
```

## See Also

- [TMacro](#)
  - [MacroByName](#)
  - [Params](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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="#">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.  |
| <a href="#">QueryRecCount</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. |

[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.

[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](#)

©

1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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 parameteri ed 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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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 parameteri ed 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.

To create a SQLInsert statement at design-time, use the query statements editor.

### See Also

- [SQL](#)
- [SQLUpdate](#)
- [SQLDelete](#)
- [SQLRefresh](#)

---

© 1997-2012 Devart. All Rights Reserved.

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 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. To create a SQLLock statement at design-time, the use query statement editor.

### See Also

- [SQL](#)
- [SQLInsert](#)
- [SQLUpdate](#)
- [SQLDelete](#)
- [SQLRefresh](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.



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.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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 [TCustomMyDataSet.FetchAll](#) = False mode. The default value of UniDirectional is False, enabling forward and backward navigation.

## See Also

- [TCustomMyDataSet.FetchAll](#)
-

© 1997-2012 Devart. All Rights Reserved.

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>  | Actualizes 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>   | 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="#">SaveSQL</a>  | Saves the SQL property value to BaseSQL.   |

[SaveToXML](#) (inherited from [TMemDataSet](#))

[SetOrderBy](#)

[SQLSaved](#)

[UnLock](#)

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

[UpdatesPending](#) (inherited from [TMemDataSet](#))

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Builds an ORDER BY clause of a SELECT statement.

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

Releases a record lock.

Frees the resources allocated for a previously prepared query on the server and client sides.

Used to indicate the update status for the current record when cached updates are enabled.

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

Used to check the status of the cached updates buffer.

Indicates the current update status for the dataset when cached updates are enabled.

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

Breaks execution of the SQL statement on the server.

## Class

[TCustomDADataset](#)

### Syntax

```
procedure BreakExec; virtual;
```

### Remarks

Call the BreakExec method to break execution of the SQL statement on the server. Execution is broken by the KILL operator execution on server. It makes sense to call BreakExec only from another thread.

© 1997-2012 Devart. All Rights Reserved.

Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.

### Class

[TCustomDADataset](#)

### 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.

© 1997-2012 Devart. All Rights Reserved.

Removes WHERE clause from the SQL property and assigns the BaseSQL property.

### Class

[TCustomDADataset](#)

### Syntax

```
procedure DeleteWhere;
```

### Remarks

Call the DeleteWhere method to remove WHERE clause from the the SQL property and assign BaseSQL.

### See Also

- [AddWhere](#)
- [BaseSQL](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.



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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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

```
MyQuery.SQL:= 'SELECT * FROM Scott.Dept ORDER BY &Order';
MyQuery.MacroByName('Order').Value:= 'DeptNo';
MyQuery.Open;
```

## See Also

- [TMacro](#)
- [Macros](#)
- [FindMacro](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

The MySQL prepared protocol has certain server restrictions, and its work is not always stable. That is why it is advisable to perform test before using preparation in production versions of applications.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

Restores the SQL property modified by AddWhere and SetOrderBy.

## Class

[TCustomDADataset](#)

## Syntax

```
procedure RestoreSQL;
```

## Remarks

Call the RestoreSQL method to restore the SQL property modified by AddWhere and SetOrderBy.

## See Also

- [AddWhere](#)
- [SetOrderBy](#)
- [SaveSQL](#)
- [SQLSaved](#)

© 1997-2012 Devart. All Rights Reserved.

Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.

## Class

[TCustomDADataset](#)

## 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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.                                      |



[LocateEx](#) (inherited from [TMemDataSet](#))

Overloaded. Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet.

[OnUpdateError](#) (inherited from [TMemDataSet](#))

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.

[Prepare](#) (inherited from [TMemDataSet](#))

Allocates resources and creates field components for a dataset.

[Prepared](#) (inherited from [TMemDataSet](#))

Determines whether a query is prepared for execution or not.

[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.

[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.

[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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
-

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

### See Also

- [AfterFetch](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.1.6 DBAccess.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.

### Inheritance Hierarchy

TObject  
    **TCustomDASQL**

© 1997-2012 Devart. All Rights Reserved.

[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.   |
| <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.   |
| <a href="#">ParamValues</a>  | Used to get or set the values of individual field parameters that are identified by name.                     |
| <a href="#">Prepared</a>     | Used to indicate whether a query is prepared for execution.   |
| <a href="#">RowsAffected</a> | Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation. |
| <a href="#">SQL</a>          | Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.     |

## Methods

| 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.   |
| <a href="#">FindParam</a>     | Finds a parameter with the specified name.  |
| <a href="#">MacroByName</a>   | Finds a Macro with the name passed in Name.   |
| <a href="#">ParamByName</a>   | Finds a parameter with the specified name.  |
| <a href="#">Prepare</a>       | Allocates, opens, and parses cursor for a query.  |
| <a href="#">UnPrepare</a>     | Frees the resources allocated for a previously prepared query on the server and client sides. |
| <a href="#">WaitExecuting</a> | Waits until TCustomDASQL executes a SQL statement.  |

## Events

| Name                         | Description                                     |
|------------------------------|---|
| <a href="#">AfterExecute</a> | Occurs after a SQL statement has been executed. |

© 1997-2012 Devart. All Rights Reserved.

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.   |
| <a href="#">ParamValues</a>  | Used to get or set the values of individual field parameters that are identified by name.                                   |
| <a href="#">Prepared</a>     | Used to indicate whether a query is prepared for execution.   |
| <a href="#">RowsAffected</a> | Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.               |
| <a href="#">SQL</a>          | Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.                   |

**See Also**

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

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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 MyDacVcl 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 MyDacVcl (MyDacClx under Kylix) unit to your project.

### See Also

- [TCustomDADataset.Debug](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
-

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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 MySQL 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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.   |
| <a href="#">FindParam</a>     | Finds a parameter with the specified name.  |
| <a href="#">MacroByName</a>   | Finds a Macro with the name passed in Name.   |
| <a href="#">ParamByName</a>   | Finds a parameter with the specified name.  |
| <a href="#">Prepare</a>       | Allocates, opens, and parses cursor for a query.  |
| <a href="#">UnPrepare</a>     | Frees the resources allocated for a previously prepared query on the server and client sides. |
| <a href="#">WaitExecuting</a> | Waits until TCustomDASQL executes a SQL statement.  |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

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 MyDAC.  |

© 1997-2012 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 is ignored.

© 1997-2012 Devart. All Rights Reserved.

Is not used in MyDAC.

## Class

[TCustomDASQL](#)

## Syntax

```
procedure Execute(Iters: integer); overload; virtual
```

### Parameters

*Iters*

Is not used in MyDAC.

## Remarks

Is not used in MyDAC.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Finds a parameter with the specified name.

### Class

[TCustomDASQL](#)

### Syntax

```
function FindParam(const Value: string): TDAParam;
```

#### 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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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

```
MyCommandSQL.Execute;  
Edit1.Text := MyCommandSQL.ParamsByName('Contact').AsString;
```

## See Also

- 

[FindParam](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

The MySQL prepared protocol has certain server restrictions, and its work is not always stable. That is why it is advisable to perform test before using preparation in production versions of applications.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

### 17.10.1.7 DBAccess.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.

## Inheritance Hierarchy

```
TObject  
  TCustomDAUpdateSQL
```

## See Also

- P:Devart.MyDac.TCustomMyDataSet.UpdateObject

© 1997-2012 Devart. All Rights Reserved.

[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.  |

[LockObject](#)

Provides ability to perform advanced adjustment of lock operations.

[LockSQL](#)

Used to lock the current record.

[ModifyObject](#)

Provides ability to perform advanced adjustment of modify operations.

[ModifySQL](#)

Used when updating a record.

[RefreshObject](#)

Provides ability to perform advanced adjustment of refresh operations.

[RefreshSQL](#)

Used to specify an SQL statement that will be used for refreshing the current record by [TCustomDADataset.RefreshRecord](#) procedure.

[SQL](#)

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.   |

© 1997-2012 Devart. All Rights Reserved.

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.     |



[ModifySQL](#)  
[RefreshObject](#)

[RefreshSQL](#)

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.

### See Also

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

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

Provides ability to perform advanced adjustment of the delete operations.

### Class

[TCustomDAUpdateSQL](#)

### Syntax

**property** DeleteObject: TComponent;

### Remarks

Assign SQL component or a TCustomMyDataSet 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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

Provides ability to perform advanced adjustment of insert operations.

### Class

[TCustomDAUpdateSQL](#)

### Syntax

```
property InsertObject: TComponent;
```

### Remarks

Assign SQL component or TCustomMyDataSet 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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

Provides ability to perform advanced adjustment of lock operations.

### Class

[TCustomDAUpdateSQL](#)

### Syntax

```
property LockObject: TComponent;
```

### Remarks

Assign a SQL component or TCustomMyDataSet 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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

Provides ability to perform advanced adjustment of modify operations.

### Class

[TCustomDAUpdateSQL](#)

### Syntax

```
property ModifyObject: TComponent;
```

### Remarks

Assign a SQL component or TCustomMyDataSet 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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

Provides ability to perform advanced adjustment of refresh operations.

### Class

[TCustomDAUpdateSQL](#)

### Syntax

```
property RefreshObject: TComponent;
```

### Remarks

Assign a SQL component or TCustomMyDataSet 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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

## 17.10.1.8 DBAccess.TDConnectionOptions Class

This class allows setting up the behaviour of the TDConnection class.

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

**Unit**

[DBAccess](#)

**Syntax**

```
TDConnectionOptions = class (TPersistent);
```

**Inheritance Hierarchy**

TObject

**TDConnectionOptions**

© 1997-2012 Devart. All Rights Reserved.

[TDConnectionOptions](#) 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. |
| <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="#">TCustomDAConnection.OnConnectionLost</a> event occurs and a failover operation can be performed after connection breaks.   |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TDConnectionOptions** class.

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

**Public**

| 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.   |

[LocalFailover](#)

If True, the [TCustomDAConnection.OnConnectionLost](#) event occurs and a failover operation can be performed after connection breaks.

**See Also**

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

© 1997-2012 Devart. All Rights Reserved.

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**

[TDACConnectionOptions](#)

**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.

© 1997-2012 Devart. All Rights Reserved.

Used to open a connection only when needed for performing a server call and closes after performing the operation.

**Class**

[TDACConnectionOptions](#)

**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.

© 1997-2012 Devart. All Rights Reserved.

Used to prevent an application from establishing a connection at the time of startup.

**Class**

[TDACConnectionOptions](#)

**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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

### 17.10.1.9 DBAccess.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);
```

## Inheritance Hierarchy

TObject

**TDADatasetOptions**

© 1997-2012 Devart. All Rights Reserved.

[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="#">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.  |
| <a href="#">QueryRecCount</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>        | Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.  |
| <a href="#">RemoveOnRefresh</a>   | Used for a dataset to locally remove a record that can not be found on the server.  |
| <a href="#">RequiredFields</a>    | Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.   |
| <a href="#">ReturnParams</a>      | Used to return the new value of fields to dataset after insert or update.   |
| <a href="#">SetFieldsReadOnly</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="#">StrictUpdate</a>      | Used for TCustomDADataset to raise an exception when the number of updated or deleted records is not equal 1.   |
| <a href="#">UpdateAllFields</a>   | Used to include all dataset fields in the generated UPDATE and INSERT statements.   |
| <a href="#">UpdateBatchSize</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.               |

© 1997-2012 Devart. All Rights Reserved.

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="#">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.  |
| <a href="#">QueryRecCount</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>        | Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.  |
| <a href="#">RemoveOnRefresh</a>   | Used for a dataset to locally remove a record that can not be found on the server.  |
| <a href="#">RequiredFields</a>    | Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.   |
| <a href="#">ReturnParams</a>      | Used to return the new value of fields to dataset after insert or update.   |
| <a href="#">SetFieldsReadOnly</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="#">StrictUpdate</a>      | Used for TCustomDADataset to raise an exception when the number of updated or deleted records is not equal 1.   |
| <a href="#">UpdateAllFields</a>   | Used to include all dataset fields in the generated UPDATE and INSERT statements.   |
| <a href="#">UpdateBatchSize</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.               |

**See Also**

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

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

Used to request default values/expressions from the server and assign them to the DefaultExpression property.

### Class

[TDADatasetOptions](#)

### Syntax

```
property DefaultValues: 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.

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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 [TCustomMyDataSet.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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.1.10 DBAccess.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.

## Inheritance Hierarchy

TObject

**TDAEncryptionOptions**

© 1997-2012 Devart. All Rights Reserved.

[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.            |

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.1.11 DBAccess.TDAMapRule Class

Class that forms 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

TObject

[TMapRule](#)

**TDAMapRule**

© 1997-2012 Devart. All Rights Reserved.

[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.   |
| <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.                               |

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

The resultant field length in Delphi.

**Class**

[TDAMapRule](#)

**Syntax**

```
property FieldLength: Integer default rlAny;
```

**Remarks**

Setting the Delphi field length after conversion.

---

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

The resultant field Scale in Delphi.

### Class

[TDAMapRule](#)

### Syntax

**property** `FieldScale: Integer default rlAny;`

### Remarks

Setting the Delphi field Scale after conversion.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.1.12 DBAccess.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

TObject  
TMapRules  
**TDAMapRules**

© 1997-2012 Devart. All Rights Reserved.

[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   |
| <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. |

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

Adding rules for mapping Database field types to Delphi field types.

## Class

[TDAMapRules](#)

## Overload List

| Name | Description |
|------|-------------|
|------|-------------|

[AddDBTypeRule\(DBType: Word; FieldType: TFieldType; IgnoreErrors: boolean\)](#)

[AddDBTypeRule\(DBType: Word; FieldType: TFieldType; FieldLength: Integer; IgnoreErrors: boolean\)](#)

[AddDBTypeRule\(DBType: Word; FieldType: TFieldType; FieldLength: Integer; FieldScale: Integer; IgnoreErrors: boolean\)](#)

[AddDBTypeRule\(DBType: Word; DBLengthMin: Integer; DBLengthMax: Integer; FieldType: TFieldType; IgnoreErrors: boolean\)](#)

[AddDBTypeRule\(DBType: Word; DBLengthMin: Integer; DBLengthMax: Integer; FieldType: TFieldType; FieldLength: Integer; IgnoreErrors: boolean\)](#)

[AddDBTypeRule\(DBType: Word; DBLengthMin: Integer; DBLengthMax: Integer; DBScaleMin: Integer; DBScaleMax: Integer; FieldType: TFieldType; IgnoreErrors: boolean\)](#)

[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.

Adding rules for mapping Database field types to Delphi field types with the specified Delphi field length.

Adding rules for mapping Database field types to Delphi field types with the specified resultant length and scale of Delphi field.

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.

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.

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.

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.

© 1997-2012 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.

© 1997-2012 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.

© 1997-2012 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.

© 1997-2012 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.

© 1997-2012 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.

© 1997-2012 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.

© 1997-2012 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.

© 1997-2012 Devart. All Rights Reserved.

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 |

© 1997-2012 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.

© 1997-2012 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.

© 1997-2012 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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

### 17.10.1.13 DBAccess.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

TObject

[TMemDataSet](#)  
**TDAMetaData**

## See Also

- [TDAMetaData.MetaDataKind](#)
- [TDAMetaData.Restrictions](#)
- [TDAMetaData.GetMetaDataKinds](#)
- [TDAMetaData.GetRestrictions](#)

© 1997-2012 Devart. All Rights Reserved.

[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. |

[GetMetaDataKinds](#)

Used to get values acceptable in the MetaDataKind property.

[GetRestrictions](#)

Used to find out which restrictions are applicable to a certain MetaDataKind.

[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.

[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.                     |

© 1997-2012 Devart. All Rights Reserved.

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.                |

[DeferredPost](#) (inherited from [TMemDataSet](#))

[GetBlob](#) (inherited from [TMemDataSet](#))

[IndexFieldNames](#) (inherited from [TMemDataSet](#))

[LocalConstraints](#) (inherited from [TMemDataSet](#))

[LocalUpdate](#) (inherited from [TMemDataSet](#))

[Locate](#) (inherited from [TMemDataSet](#))

[LocateEx](#) (inherited from [TMemDataSet](#))

[MetaDataKind](#)

[OnUpdateError](#) (inherited from [TMemDataSet](#))

[OnUpdateRecord](#) (inherited from [TMemDataSet](#))

[Prepare](#) (inherited from [TMemDataSet](#))

[Prepared](#) (inherited from [TMemDataSet](#))

[RestoreUpdates](#) (inherited from [TMemDataSet](#))

[Restrictions](#)

[RevertRecord](#) (inherited from [TMemDataSet](#))

[SaveToXML](#) (inherited from [TMemDataSet](#))

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

[UpdatesPending](#) (inherited from [TMemDataSet](#))

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Makes permanent changes to the database server.

Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.

Used to get or set the list of fields on which the recordset is sorted.

Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

Used to prevent implicit update of rows on database server.

Overloaded. Searches a dataset for a specific record and positions the cursor on it.

Overloaded. Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet.

Used to specify which kind of metainformation to show.

Occurs when an exception is generated while cached updates are applied to a database.

Occurs when a single update component can not handle the updates.

Allocates resources and creates field components for a dataset.

Determines whether a query is prepared for execution or not.

Marks all records in the cache of updates as unapplied.

Used to provide one or more conditions restricting the list of objects to be described.

Cancels changes made to the current record when cached updates are enabled.

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Frees the resources allocated for a previously prepared query on the server and client sides.

Used to indicate the update status for the current record when cached updates are enabled.

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

Used to check the status of the cached updates buffer.

Indicates the current update status for the dataset when cached updates are enabled.

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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:

| MetaDataKind        | Description   |
|---------------------|---|
| Columns             | show metainformation about columns of existing tables   |
| Constraints         | show metainformation about the constraints defined in the database  |
| Databases           | show metainformation about existing databases   |
| 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 MetadataKind 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 MetaDataKind                 |
| Tables              | show metainformation about existing tables  |

If you provide a value that equals neither of the values described in the table, an error will be raised.

## See Also

- [Restrictions](#)

© 1997-2012 Devart. All Rights Reserved.



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](#)

© 1997-2012 Devart. All Rights Reserved.

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. |

[OnUpdateError](#) (inherited from [TMemDataSet](#))

[OnUpdateRecord](#) (inherited from [TMemDataSet](#))

[Prepare](#) (inherited from [TMemDataSet](#))

[Prepared](#) (inherited from [TMemDataSet](#))

[RestoreUpdates](#) (inherited from [TMemDataSet](#))

[RevertRecord](#) (inherited from [TMemDataSet](#))

[SaveToXML](#) (inherited from [TMemDataSet](#))

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

[UpdatesPending](#) (inherited from [TMemDataSet](#))

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Occurs when an exception is generated while cached updates are applied to a database.

Occurs when a single update component can not handle the updates.

Allocates resources and creates field components for a dataset.

Determines whether a query is prepared for execution or not.

Marks all records in the cache of updates as unapplied.

Cancels changes made to the current record when cached updates are enabled.

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Frees the resources allocated for a previously prepared query on the server and client sides.

Used to indicate the update status for the current record when cached updates are enabled.

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

Used to check the status of the cached updates buffer.

Indicates the current update status for the dataset when cached updates are enabled.

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

### 17.10.1.14 DBAccess.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.

## Inheritance Hierarchy

```
TObject
  TDAParam
```

## See Also

- [TCustomDADataset](#)
- [TCustomDASQL](#)
- [TDAParams](#)

© 1997-2012 Devart. All Rights Reserved.

[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.         |
| <a href="#">SetBlobData</a>      | Overloaded. Writes the data from a specified buffer to BLOB.     |

© 1997-2012 Devart. All Rights Reserved.

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  |
|---------------------------|--|
| <a href="#">DataType</a>  | Indicates the data type of the parameter.                |
| <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. |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---



© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

Writes the data from a specified buffer to BLOB.

## Class

[TDAParam](#)

## Overload List

| Name  | Description                                      |
|---|--|
| <a href="#">SetBlobData</a>                       | Writes the data from a specified buffer to BLOB. |
| <a href="#">SetBlobData(Buffer: TValueBuffer)</a> | Writes the data from a specified buffer to BLOB. |

---

© 1997-2012 Devart. All Rights Reserved.

Writes the data from a specified buffer to BLOB.

## Unit

## Syntax

## Remarks

Call the SetBlobData method to write data from a specified buffer to BLOB.

---

© 1997-2012 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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.1.15 DBAccess.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.

### Inheritance Hierarchy

TObject  
    **TDAParams**

### See Also

- [TCustomDADataset.Params](#)
  - [TCustomDASQL.Params](#)
  - [TDAParam](#)
- 

© 1997-2012 Devart. All Rights Reserved.

[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. |

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

### 17.10.1.16 DBAccess.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.

## Inheritance Hierarchy

TObject

**TDATransaction**

© 1997-2012 Devart. All Rights Reserved.

[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. |

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.



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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

## 17.10.1.17 DBAccess.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.

**Inheritance Hierarchy**

TObject  
    **TMacro**

**See Also**

- [TMacros](#)

---

© 1997-2012 Devart. All Rights Reserved.

[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.          |
| <a href="#">AsString</a>   | Used to assign the string value to a macro.        |
| <a href="#">Name</a>       | Used to identify a particular macro.               |
| <a href="#">Value</a>      | Used to set the value to a macro.                  |

---

© 1997-2012 Devart. All Rights Reserved.

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.   |

[AsString](#)

Used to assign the string value to a macro.

## Published

### Name

[Active](#)

[Name](#)

[Value](#)

### Description

Used to determine if the macro should be expanded.

Used to identify a particular macro.

Used to set the value to a macro.

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

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

```
MyQuery.SQL.Text := 'SELECT * FROM Dept WHERE DeptNo > 20 &Cond1';
MyQuery.Macros[0].Value := 'and DName is NULL';
MyQuery.Macros[0].Active:= False;
```

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

Used to identify a particular macro.

**Class**

[TMacro](#)

**Syntax**

```
property Name: string;
```

**Remarks**

Use the Name property to identify a particular macro.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

## 17.10.1.18 DBAccess.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.

**Inheritance Hierarchy**

TObject  
  **TMacros**

**See Also**

- [TMacro](#)

© 1997-2012 Devart. All Rights Reserved.

[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.                |
| <a href="#">Scan</a>         | Creates a macros from the passed SQL statement.                    |

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.    |
| <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.                |
| <a href="#">Scan</a>         | Creates a macros from the passed SQL statement.                    |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.1.19 DBAccess.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);
```

### Inheritance Hierarchy

TObject

**TPoolingOptions**

© 1997-2012 Devart. All Rights Reserved.

[TPoolingOptions](#) class overview.

### Properties

| 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.       |



[Validate](#)

Used for a connection to be validated when it is returned from the pool.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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 TCustomDACConnection component.

---

© 1997-2012 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> .   |
| <a href="#">TUpdateExecuteEvent</a>      | This type is used for the <a href="#">TCustomDADataset.AfterUpdateExecute</a> and <a href="#">TCustomDADataset.BeforeUpdateExecute</a> events. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.2.1 DBAccess.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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.2.2 DBAccess.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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.2.3 DBAccess.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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.2.4 DBAccess.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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.2.5 DBAccess.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 .

© 1997-2012 Devart. All Rights Reserved.

## 17.10.2.6 DBAccess.TDATransactionErrorEvent Procedure Reference

This type is used for the [TDATransaction.OnError](#) event.

**Unit**

[DBAccess](#)

**Syntax**

```
TDATransactionErrorEvent = procedure (Sender: TObject; E: EDAError
; 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.

© 1997-2012 Devart. All Rights Reserved.

## 17.10.2.7 DBAccess.TRefreshOptions Set

Represents the set of [TRefreshOption](#).

**Unit**

[DBAccess](#)

**Syntax**

```
TRefreshOptions = set of TRefreshOption;
```

© 1997-2012 Devart. All Rights Reserved.

## 17.10.2.8 DBAccess.TUpdateExecuteEvent Procedure Reference

This type is used for the TCustomDADataset.AfterUpdateExecute and TCustomDADataset.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.

---

### 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>      | This enumeration defines a type of an editing record locking. |
| <a href="#">TRefreshOption</a> | Indicates when the editing record will be refreshed.          |
| <a href="#">TRetryMode</a>     | Specifies the application behavior when connection is lost.   |

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.3.1 DBAccess.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>lsCustom</b>     | Set the language of labels in the connect dialog manually.      |
| <b>lsEnglish</b>    | Set English as the language of labels in the connect dialog.    |
| <b>lsFrench</b>     | Set French as the language of labels in the connect dialog.     |
| <b>lsGerman</b>     | Set German as the language of labels in the connect dialog.     |
| <b>lsItalian</b>    | Set Italian as the language of labels in the connect dialog.    |
| <b>lsPolish</b>     | Set Polish as the language of labels in the connect dialog.     |
| <b>lsPortuguese</b> | Set Portuguese as the language of labels in the connect dialog. |
| <b>lsRussian</b>    | Set Russian as the language of labels in the connect dialog.    |
| <b>lsSpanish</b>    | Set Spanish as the language of labels in the connect dialog.    |

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.3.2 DBAccess.TLockMode Enumeration

This enumeration defines a type of an editing record locking.

##### Unit

[DBAccess](#)

##### Syntax

```
TLockMode = (lmNone, lmPessimistic, lmOptimistic);
```

##### Values

| Value               | Meaning   |
|---------------------|---|
| <b>lmNone</b>       | No locking is performed. This should only be used in single user applications. The default value.   |
| <b>lmOptimistic</b> | Locking is performed when user posts an edited record. After this the lock is released. Locking is performed by the RefreshRecord method. |

**ImPessimistic**

Locking is performed when the user starts editing a record. The lock remains until the user posts or cancels the changes.

---

© 1997-2012 Devart. All Rights Reserved.

## 17.10.3.3 DBAccess.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.  |

---

© 1997-2012 Devart. All Rights Reserved.

## 17.10.3.4 DBAccess.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. |

---

© 1997-2012 Devart. All Rights Reserved.



## 17.10.4 Variables

Variables in the **DBAccess** unit.

### Variables

| Name                                      | Description  |
|---|--|
| <a href="#">BaseSQLOldBehavior</a>        | After assigning SQL text and modifying it by <a href="#">AddWhere</a> , <a href="#">DeleteWhere</a> , and <a href="#">SetOrderBy</a> , all subsequent changes of the SQL property will not be reflected in the BaseSQL property. |
| <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.  |
| <a href="#">SQLGeneratorCompatibility</a> | The value of the <a href="#">TCustomDADDataSet.BaseSQL</a> property is used to complete the refresh SQL statement, if the manually assigned <a href="#">TCustomDAUpdateSQL.RefreshSQL</a> property contains only WHERE clause.   |

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.4.1 DBAccess.BaseSQLOldBehavior Variable

After assigning SQL text and modifying it by [AddWhere](#), [DeleteWhere](#), and [SetOrderBy](#), all subsequent changes of the SQL property will not be reflected in the BaseSQL property.

### Unit

[DBAccess](#)

### Syntax

```
BaseSQLOldBehavior: boolean;
```

### Remarks

The [BaseSQL](#) property is similar to the SQL property, but it does not store changes made by the [AddWhere](#), [DeleteWhere](#), and [SetOrderBy](#) methods. After assigning SQL text and modifying it by one of these methods, all subsequent changes of the SQL property will not be reflected in the BaseSQL property. This behavior was changed in MyDAC 4.00.2.8. To restore old behavior, set the BaseSQLOldBehavior variable to True.

© 1997-2012 Devart. All Rights Reserved.

#### 17.10.4.2 DBAccess.ChangeCursor Variable

When set to True allows data access components to change screen cursor for the execution time.

### Unit

[DBAccess](#)

### Syntax

```
ChangeCursor: boolean;
```

© 1997-2012 Devart. All Rights Reserved.

## 17.10.4.3 DBAccess.MacroChar Variable

Determinates what character is used for macros.

**Unit**

[DBAccess](#)

**Syntax**

```
MacroChar: _char;
```

---

© 1997-2012 Devart. All Rights Reserved.

## 17.10.4.4 DBAccess.SQLGeneratorCompatibility Variable

The value of the [TCustomDADataset.BaseSQL](#) property is used to complete the refresh SQL statement, if the manually assigned [TCustomDAUpdateSQL.RefreshSQL](#) property contains only WHERE clause.

**Unit**

[DBAccess](#)

**Syntax**

```
SQLGeneratorCompatibility: boolean;
```

**Remarks**

If the manually assigned [TCustomDAUpdateSQL.RefreshSQL](#) property contains only WHERE clause, MyDAC uses the value of the [TCustomDADataset.BaseSQL](#) property to complete the refresh SQL statement. In this situation all modifications applied to the SELECT query by functions [TCustomDADataset.AddWhere](#), [TCustomDADataset.DeleteWhere](#) are not taken into account. This behavior was changed in MyDAC 5.00.0.4. To restore the old behavior, set the BaseSQLOldBehavior variable to True.

---

© 1997-2012 Devart. All Rights Reserved.

## **17.11 Devart.Dac.DataAdapter**

This unit contains implementation of the DADDataAdapter class.

### **Classes**

| <b>Name</b>                    | <b>Description</b>  |
|--------------------------------|---|
| <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. |

---

© 1997-2012 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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.11.1.1 Devart.Dac.DataAdapter.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.

### Inheritance Hierarchy

TObject  
**DADDataAdapter**

© 1997-2012 Devart. All Rights Reserved.

[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. |

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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 DADataAdapter 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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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**

[DADataAdapter](#)

**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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

## **17.12 Devart.MyDac.DataAdapter**

This unit contains implementation of the MyDataAdapter class.

### **Classes**

| <b>Name</b>                   | <b>Description</b>  |
|-------------------------------|---|
| <a href="#">MyDataAdapter</a> | A class for using with <a href="#">TCustomMyDataSet</a> components and as data source for retrieving and saving data. |

---

© 1997-2012 Devart. All Rights Reserved.

## 17.12.1 Classes

Classes in the **Devart.MyDac.DataAdapter** unit.

### Classes

| Name                          | Description   |
|-------------------------------|---|
| <a href="#">MyDataAdapter</a> | A class for using with <a href="#">TCustomMyDataSet</a> components and as data source for retrieving and saving data. |

---

© 1997-2012 Devart. All Rights Reserved.

#### 17.12.1.1 Devart.MyDac.DataAdapter.MyDataAdapter Class

A class for using with [TCustomMyDataSet](#) components and as data source for retrieving and saving data. For a list of all members of this type, see [MyDataAdapter](#) members.

### Unit

[Devart.MyDac.DataAdapter](#)

### Syntax

```
MyDataAdapter = class (DADDataAdapter) ;
```

### Remarks

The MyDataAdapter class is designed for using with [TCustomMyDataSet](#) components and as data source for retrieving and saving data. MyDataAdapter 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

TObject  
    [DADDataAdapter](#)  
        **MyDataAdapter**

### See Also

- [DADDataAdapter](#)
- 

© 1997-2012 Devart. All Rights Reserved.

[MyDataAdapter](#) 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.

---

## 17.13 MemData

This unit contains classes for storing data in memory.

### Classes

| Name                            | Description  |
|---------------------------------|--|
| <a href="#">TAttribute</a>      | TAttribute is not used in MyDAC.   |
| <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.                  |
| <a href="#">TObjectType</a>     | This class is not used.  |
| <a href="#">TSharedObject</a>   | 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="#">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.                |

---

### 17.13.1 Classes

Classes in the **MemData** unit.

#### Classes

| Name                            | Description  |
|---------------------------------|--|
| <a href="#">TAttribute</a>      | TAttribute is not used in MyDAC.   |
| <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.                  |
| <a href="#">TObjectType</a>     | This class is not used.  |
| <a href="#">TSharedObject</a>   | A base class that allows to simplify memory management for object referenced by several other objects. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.13.1.1 MemData.TAttribute Class

TAttribute is not used in MyDAC.

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

#### Unit

[MemData](#)

#### Syntax

```
TAttribute = class (System.TObject);
```

#### Inheritance Hierarchy

```
TObject
  TAttribute
```

© 1997-2012 Devart. All Rights Reserved.

[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.                          |

[Scale](#)

Returns the scale of dtFloat and dtInteger attributes.

[Si e](#)

Returns the si e of an attribute value in external representation.

© 1997-2012 Devart. All Rights Reserved.

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="#">DataSi e</a>    | Returns the si e 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="#">Si e</a>        | Returns the si e of an attribute value in external representation.   |

**See Also**

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

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

Returns an offset of the attribute value in internal representation.

### Class

[TAttribute](#)

### Syntax

**property** Offset: Integer;

### Remarks

Use the DataSi e property to learn an offset of the attribute value in internal representation.

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

Returns the si e of an attribute value in external representation.

### Class

[TAttribute](#)

### Syntax

**property** Size: Integer;

### Remarks

Read Si e to learn the si e of an attribute value in external representation.  
For example:

|         |                           |
|---------|---------------------------|
| dtDate  | 8 (si eof<br>(TDateTime)) |
| dtFloat | 8 (si eof(Double))        |

dtInteger                      4 (si eof(Integer))

## See Also

- [DataSieve](#)

© 1997-2012 Devart. All Rights Reserved.

### 17.13.1.2 MemData.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

```
TObject
  TSharedObject
    TBlob
```

## See Also

- TBlob in Delphi Help
- [TMemDataSet.GetBlob](#)

© 1997-2012 Devart. All Rights Reserved.

[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.               |
| <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.              |
| <a href="#">Write</a>   | Stores a raw sequence of bytes into a TBlob object.             |

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.   |
| <a href="#">Write</a>  | Stores a raw sequence of bytes into a TBlob object.  |

### See Also

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

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

### 17.13.1.3 MemData.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, Borland Delphi 8 (for .NET) 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: TCompressedBlob) : Integer;
TUncompressProc = function (dest: IntPtr; destLen: IntPtr; source: TCompressedBlob) : Integer;
```

**var**

```
CompressProc: TCompressProc;
UncompressProc: TUncompressProc;
```

## Inheritance Hierarchy

TObject

[TSharedObject](#)

[TBlob](#)

**TCompressedBlob**

## See Also

- [TBlob](#)
- [TMemDataSet.GetBlob](#)
- [TCustomDADataset.Options](#)

© 1997-2012 Devart. All Rights Reserved.

[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. |

[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.  |

© 1997-2012 Devart. All Rights Reserved.

### 17.13.1.4 MemData.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

```
TObject
  TSharedObject
    TDBObject
```

© 1997-2012 Devart. All Rights Reserved.

[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.  |

© 1997-2012 Devart. All Rights Reserved.

### 17.13.1.5 MemData.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

```
TObject
  TSharedObject
    TObjectType
```

© 1997-2012 Devart. All Rights Reserved.

[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.                               |
| <a href="#">DataType</a>   | Used to indicate the type of object dtObject, dtArray or dtTable. |
| <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 an object instance.                     |

## 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="#">AttributeByName</a>  | 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.

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TObjectType** class.

For a complete list of the **TObjectType** class members, see the [TObjectType 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="#">AttributeCount</a>   | Used to indicate the number of attributes of type.   |
| <a href="#">Attributes</a>   | Used to access separate attributes.  |
| <a href="#">DataType</a>   | Used to indicate the type of object dtObject, dtArray or dtTable.                                  |
| <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 an object instance.  |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

Methods of the **TObjectType** class.  
For a complete list of the **TObjectType** class members, see the [TObjectType 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 <code>TSharedObject</code> object. |
| <a href="#">AttributeByName</a>  | 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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

### 17.13.1.6 MemData.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.

## Inheritance Hierarchy

```
TObject
  TSharedObject
```

## See Also

- [TBlob](#)
- [TObjectType](#)

© 1997-2012 Devart. All Rights Reserved.

[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.  |

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
-

## 17.13.2 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.                   |

© 1997-2012 Devart. All Rights Reserved.

#### 17.13.2.1 MemData.TLocateExOptions Set

Represents the set of [TLocateExOption](#).

### Unit

[MemData](#)

### Syntax

```
TLocateExOptions = set of TLocateExOption;
```

© 1997-2012 Devart. All Rights Reserved.

#### 17.13.2.2 MemData.TUpdateRecKinds Set

Represents the set of TUpdateRecKind.

### Unit

[MemData](#)

### Syntax

```
TUpdateRecKinds = set of TUpdateRecKind;
```

© 1997-2012 Devart. All Rights Reserved.

### 17.13.3 Enumerations

Enumerations in the **MemData** unit.

#### Enumerations

| Name                            | Description   |
|---------------------------------|---|
| <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.                |

© 1997-2012 Devart. All Rights Reserved.

#### 17.13.3.1 MemData.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.   |

© 1997-2012 Devart. All Rights Reserved.



## 17.13.3.2 MemData.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 on client is in TBCD format and is represented as TFMTBCDField. Allows to reflect the whole range of possible values for the MySQL NUMERIC type without accuracy losses. Fields of this type are processed quite slowly. Not supported for Delphi 5 and C++Builder 5. |

© 1997-2012 Devart. All Rights Reserved.

## 17.13.3.3 MemData.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.   |

© 1997-2012 Devart. All Rights Reserved.

## 17.13.3.4 MemData.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.  |

---

© 1997-2012 Devart. All Rights Reserved.

#### 17.13.3.5 MemData.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.  |

---

© 1997-2012 Devart. All Rights Reserved.

## 17.14 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. |

## 17.14.1 Classes

Classes in the **MemDS** unit.

### Classes

| Name                        | Description   |
|-----------------------------|---|
| <a href="#">TMemDataSet</a> | A base class for working with data and manipulating data in memory. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.14.1.1 MemDS.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.

### Inheritance Hierarchy

```
TObject
  TMemDataSet
```

© 1997-2012 Devart. All Rights Reserved.

[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.                                |
| <a href="#">UpdatesPending</a>    | 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. |

[CancelUpdates](#)

Clears all pending cached updates from cache and restores dataset in its prior state.

[CommitUpdates](#)[DeferredPost](#)

Clears the cached updates buffer. Makes permanent changes to the database server.

[GetBlob](#)

Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.

[Locate](#)

Overloaded. Searches a dataset for a specific record and positions the cursor on it.

[LocateEx](#)

Overloaded. Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet.

[Prepare](#)

Allocates resources and creates field components for a dataset.

[RestoreUpdates](#)

Marks all records in the cache of updates as unapplied.

[RevertRecord](#)

Cancels changes made to the current record when cached updates are enabled.

[SaveToXML](#)

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

[UnPrepare](#)

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.

[UpdateStatus](#)

Indicates the current update status for the dataset when cached updates are enabled.

## 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.                     |

© 1997-2012 Devart. All Rights Reserved.

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. |

[LocalConstraints](#)

Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

[LocalUpdate](#)

Used to prevent implicit update of rows on database server.

[Prepared](#)

Determines whether a query is prepared for execution or not.

[UpdateRecordTypes](#)

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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';
```

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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. The Prepared property currently is not supported by MySQL and is always False.

### See Also

- [Prepare](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

Methods of the **TMemDataSet** class.

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

## Public

| 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><br><a href="#">DeferredPost</a> | Clears the cached updates buffer. 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 ApplyUpdates method while cached updates are enabled.                           |
| <a href="#">UpdateStatus</a>                                  | Indicates the current update status for the dataset when cached updates are enabled.                                       |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

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. |

© 1997-2012 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.

## Example

The following procedure illustrates how to apply a dataset's cached updates to a database in response to a button click:

```
procedure ApplyButtonClick(Sender: TObject);
begin
    with MyQuery do
        begin
            Session.StartTransaction;
            try
                ... {Modify data}
                ApplyUpdates; {try to write the updates to the database}
                Session.Commit; {on success, commit the changes}
            except
                RestoreUpdates; {restore update result for applied records}
                Session.Rollback; {on failure, undo the changes}
                raise; {raise the exception to prevent a call to CommitUpdates!}
            end;
            CommitUpdates; {on success, clear the cache}
        end;
end;
```

## See Also

- [TMemDataSet.CachedUpdates](#)
- [TMemDataSet.CancelUpdates](#)
- [TMemDataSet.CommitUpdates](#)
- [TMemDataSet.UpdateStatus](#)

---

© 1997-2012 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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)
-

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.         |

---

© 1997-2012 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.

© 1997-2012 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

```
MyQuery1.GetBlob('Comment').SaveToFile('Comment.txt');
```

## See Also

- 

[TBlob](#)

© 1997-2012 Devart. All Rights Reserved.

Searches a dataset for a specific record and positions the cursor on it.

## Class

[TMemDataSet](#)

## Overload List

| Name   | Description  |
|--|--|
| <a href="#">Locate</a> ( <b>const</b> KeyFields: <b>array of</b> TField; <b>const</b> KeyValues: <b>variant</b> ; Options: TLocateOptions) | Searches a dataset by the specified fields for a specific record and positions cursor on it. |

[Locate\(const KeyFields: string; const KeyValues: variant; Options: TLocateOptions\)](#)

Searches a dataset by the fields specified by name for a specific record and positions the cursor on it.

© 1997-2012 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.

© 1997-2012 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](#)

© 1997-2012 Devart. All Rights Reserved.

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.      |
| <a href="#">LocateEx(const KeyFields: string; 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 field names. |

© 1997-2012 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.

© 1997-2012 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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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 MySQL prepared protocol has certain server restrictions, and its work is not always stable. That is why it is advisable to perform test before using preparation in production versions of applications.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.   |

---

© 1997-2012 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](#)
- 

© 1997-2012 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.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

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](#)

© 1997-2012 Devart. All Rights Reserved.

## 17.14.2 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 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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.14.2.1 MemDS.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 MyDAC 5.20.0.12, 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.

© 1997-2012 Devart. All Rights Reserved.

#### 17.14.2.2 MemDS.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 MyDAC 5.20.0.11, 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.

© 1997-2012 Devart. All Rights Reserved.

## 17.15 MyAccess

This unit contains implementation of most public classes of MyDAC.

### Classes

| Name  | Description  |
|---|--|
| <a href="#"><u>TCustomMyConnection</u></a>        | A base class for connecting to MySQL server.   |
| <a href="#"><u>TCustomMyConnectionOptions</u></a> | This class allows setting up the behaviour of the TCustomMyConnection class.   |
| <a href="#"><u>TCustomMyDataSet</u></a>           | A base class defining functionality for the classes derived from it.   |
| <a href="#"><u>TCustomMyStoredProc</u></a>        | A class implementing functionality to access stored procedures on a database server.   |
| <a href="#"><u>TCustomMyTable</u></a>             | A base class that defines functionality for descendant classes which access data in a single table without writing SQL statements. |
| <a href="#"><u>TMyCommand</u></a>                 | A component for execution of SQL statements and stored procedures which do not return rowsets.                                     |
| <a href="#"><u>TMyConnection</u></a>              | A component for setting up and controlling connections to MySQL database server.   |
| <a href="#"><u>TMyConnectionOptions</u></a>       | This class allows setting up the behaviour of the TMyConnection class.   |
| <a href="#"><u>TMyConnectionSSLOptions</u></a>    | This class allows setting up the behaviour of the TMyConnection class.   |
| <a href="#"><u>TMyDataSetOptions</u></a>          | This class allows setting up the behaviour of the TMyDataSet class.  |
| <a href="#"><u>TMyDataSource</u></a>              | TMyDataSource provides an interface between a MyDAC dataset components and data-aware controls on a form.                          |
| <a href="#"><u>TMyEncryptor</u></a>               | The class that performs encrypting and decrypting of data.   |
| <a href="#"><u>TMyMetaData</u></a>                | A component for obtaining metainformation about database objects from the server.  |
| <a href="#"><u>TMyQuery</u></a>                   | A component for executing queries and operating record sets. It also provides flexible way to update data.                         |
| <a href="#"><u>TMyStoredProc</u></a>              | A component for accessing and executing stored procedures and functions.   |
| <a href="#"><u>TMyTable</u></a>                   | A component for retrieving and updating data in a single table without writing SQL statements.                                     |
| <a href="#"><u>TMyTableOptions</u></a>            | This class allows setting up the behaviour of the TMyTable class.  |
| <a href="#"><u>TMyTransaction</u></a>             | A component for managing transactions.   |

[TMyUpdateSQL](#)

A component for tuning update operations for the DataSet component.

## Types

| Name                                  | Description  |
|---------------------------------------|--|
| <a href="#">TMyUpdateExecuteEvent</a> | This type is used for the E:Devart.MyDac.TCustomMyDataSet.AfterUpdateExecute and E:Devart.MyDac.TCustomMyDataSet.BeforeUpdateExecute events. |

## Enumerations

| Name                              | Description  |
|-----------------------------------|--|
| <a href="#">TLockRecordType</a>   | Specifies the type of the record locking.  |
| <a href="#">TLockType</a>         | Specifies the type of the table locking.   |
| <a href="#">TMyIsolationLevel</a> | Specifies the extent to which all outside transactions interfere with subsequent transactions of current connection. |

## Routines

| Name                          | Description   |
|-------------------------------|---|
| <a href="#">GetServerList</a> | Returns the list of the MySQL servers in LAN. MySQL server does not provide usual ways of such list getting, so it can be incomplete. |

## Constants

| Name                         | Description   |
|------------------------------|---|
| <a href="#">MydacVersion</a> | Read this constant to get current version number for MyDAC. |

---



### 17.15.1 Classes

Classes in the **MyAccess** unit.

#### Classes

| Name  | Description  |
|---|--|
| <a href="#"><u>TCustomMyConnection</u></a>        | A base class for connecting to MySQL server.   |
| <a href="#"><u>TCustomMyConnectionOptions</u></a> | This class allows setting up the behaviour of the TCustomMyConnection class.   |
| <a href="#"><u>TCustomMyDataSet</u></a>           | A base class defining functionality for the classes derived from it.   |
| <a href="#"><u>TCustomMyStoredProc</u></a>        | A class implementing functionality to access stored procedures on a database server.   |
| <a href="#"><u>TCustomMyTable</u></a>             | A base class that defines functionality for descendant classes which access data in a single table without writing SQL statements. |
| <a href="#"><u>TMyCommand</u></a>                 | A component for execution of SQL statements and stored procedures which do not return rowsets.                                     |
| <a href="#"><u>TMyConnection</u></a>              | A component for setting up and controlling connections to MySQL database server.   |
| <a href="#"><u>TMyConnectionOptions</u></a>       | This class allows setting up the behaviour of the TMyConnection class.   |
| <a href="#"><u>TMyConnectionSSLOptions</u></a>    | This class allows setting up the behaviour of the TMyConnection class.   |
| <a href="#"><u>TMyDataSetOptions</u></a>          | This class allows setting up the behaviour of the TMyDataSet class.  |
| <a href="#"><u>TMyDataSource</u></a>              | TMyDataSource provides an interface between a MyDAC dataset components and data-aware controls on a form.                          |
| <a href="#"><u>TMyEncryptor</u></a>               | The class that performs encrypting and decrypting of data.   |
| <a href="#"><u>TMyMetaData</u></a>                | A component for obtaining metainformation about database objects from the server.  |
| <a href="#"><u>TMyQuery</u></a>                   | A component for executing queries and operating record sets. It also provides flexible way to update data.                         |
| <a href="#"><u>TMyStoredProc</u></a>              | A component for accessing and executing stored procedures and functions.   |
| <a href="#"><u>TMyTable</u></a>                   | A component for retrieving and updating data in a single table without writing SQL statements.                                     |
| <a href="#"><u>TMyTableOptions</u></a>            | This class allows setting up the behaviour of the TMyTable class.  |
| <a href="#"><u>TMyTransaction</u></a>             | A component for managing transactions.   |
| <a href="#"><u>TMyUpdateSQL</u></a>               | A component for tuning update operations for the DataSet component.  |

© 1997-2012 Devart. All Rights Reserved.

#### 17.15.1.1 MyAccess.TCustomMyConnection Class

A base class for connecting to MySQL server.

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

### Unit

[MyAccess](#)

### Syntax

```
TCustomMyConnection = class(TCustomDAConnection) ;
```

### Remarks

The TCustomMyConnection component is used to establish connection to database server, provide customi ed login support, and perform transaction control. TCustomMyConnection is the base component for connecting to MySQL server.

### Inheritance Hierarchy

TObject

[TCustomDAConnection](#)

**TCustomMyConnection**

### See Also

- [TMyConnection](#)
- [TMyEmbConnection](#)

© 1997-2012 Devart. All Rights Reserved.

[TCustomMyConnection](#) class overview.

### Properties

| Name  | Description  |
|---|--|
| <a href="#">ClientVersion</a>   | Contains the version of the MySQL Client library.  |
| <a href="#">ConnectDialog</a> (inherited from <a href="#">TCustomDAConnection</a> ) | Allows to link a <a href="#">TCustomConnectDialog</a> component.   |
| <a href="#">ConnectionTimeout</a>   | Used to specify the amount of time to attempt to establish a connection.   |
| <a href="#">ConvertEOL</a> (inherited from <a href="#">TCustomDAConnection</a> )    | Allows customi ing line breaks in string fields and parameters.  |
| <a href="#">Database</a>  | Used to specify a database name that is a default source of data for SQL queries once a connection is established.     |
| <a href="#">InTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> ) | Indicates whether the transaction is active.   |
| <a href="#">IsolationLevel</a>  | Used to specify extent to which all outside transactions interfere with subsequent transactions of current connection. |
| <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>   | Specifies the behaviour of the TMyConnectionOptions object.  |
| <a href="#">Password</a> (inherited from <a href="#">TCustomDAConnection</a> )      | Serves to supply a password for login.   |

[Pooling](#) (inherited from [TCustomDAConnection](#))

[PoolingOptions](#) (inherited from [TCustomDAConnection](#))

[Server](#) (inherited from [TCustomDAConnection](#))

[ServerVersion](#)

[ThreadId](#)

[Username](#) (inherited from [TCustomDAConnection](#))

Enables or disables using connection pool.

Specifies the behaviour of connection pool.

Serves to supply the server name for login.

Holds the version of MySQL server.

Used to return the thread ID of the current connection.

Used to supply a user name for login.

## Methods

| Name   | Description   |
|--|---|
| <a href="#">ApplyUpdates</a> (inherited from <a href="#">TCustomDAConnection</a> )       | Overloaded. Applies changes in datasets.  |
| <a href="#">AssignConnect</a>  | Shares database connection between the TCustomMyConnection components.                                  |
| <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="#">CreateDataSet</a>  | Returns a new instance of TCustomMyDataSet class and associates it with this connection object.         |
| <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="#">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>  | Executes any SQL statement outside <a href="#">TMyQuery</a> or <a href="#">TMyCommand</a> components.   |
| <a href="#">ExecSQLEx</a> (inherited from <a href="#">TCustomDAConnection</a> )          | Executes any SQL statement outside the TQuery or TSQL components.                                       |
| <a href="#">GetCharsetNames</a>  | Populates a string list with the names of available charsets.   |
| <a href="#">GetDatabaseNames</a> (inherited from <a href="#">TCustomDAConnection</a> )   | Returns a database list from the server.  |
| <a href="#">GetExecuteInfo</a>   | Returns the result of the last query execution.   |
| <a href="#">GetStoredProcNames</a> (inherited from <a href="#">TCustomDAConnection</a> ) | Returns a list of stored procedures from the server.  |
| <a href="#">GetTriggerNames</a>  | Returns a list of triggers from the server.   |
| <a href="#">MonitorMessage</a> (inherited from <a href="#">TCustomDAConnection</a> )     | Sends a specified message through the <a href="#">TCustomDASQLMonitor</a> component.                    |
| <a href="#">Ping</a>   | Allows to avoid automatic disconnection of the client by the server.                                    |
| <a href="#">ReleaseSavepoint</a>   | Releases the specified savepoint without affecting any work that has been performed after its creation. |

|  |  |
|--|--|
| <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="#">RollbackToSavepoint</a>  | Cancels all updates for the current transaction.                               |
| <a href="#">Savepoint</a>  | Defines a point in the transaction to which you can roll back later.           |
| <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.                   |
| <a href="#">OnError</a> (inherited from <a href="#">TCustomDAConnection</a> )          | This event occurs when an error has arisen in the connection. |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TCustomMyConnection** class.

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

## Public

| Name  | Description  |
|---|--|
| <a href="#">ApplyUpdates</a> (inherited from <a href="#">TCustomDAConnection</a> )  | Overloaded. Applies changes in datasets.   |
| <a href="#">ClientVersion</a>   | Contains the version of the MySQL Client library.  |
| <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="#">ConnectionTimeout</a>   | Used to specify the amount of time to attempt to establish a connection.   |
| <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="#">Database</a>  | Used to specify a database name that is a default source of data for SQL queries once a connection is established. |
| <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="#">InTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )      | Indicates whether the transaction is active.   |
| <a href="#">IsolationLevel</a>   | Used to specify extent to which all outside transactions interfere with subsequent transactions of current connection. |
| <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="#">Options</a>  | Specifies the behaviour of the TMyConnectionOptions object.  |
| <a href="#">Password</a> (inherited from <a href="#">TCustomDAConnection</a> )           | Serves to supply a password for login.   |
| <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="#">Server</a> (inherited from <a href="#">TCustomDAConnection</a> )             | Serves to supply the server name for login.  |
| <a href="#">ServerVersion</a>  | Holds the version of MySQL server.   |
| <a href="#">StartTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )   | Begins a new user transaction.   |
| <a href="#">ThreadId</a>   | Used to return the thread ID of the current connection.  |
| <a href="#">Username</a> (inherited from <a href="#">TCustomDAConnection</a> )           | Used to supply a user name for login.  |

### See Also

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

© 1997-2012 Devart. All Rights Reserved.

Contains the version of the MySQL Client library.

### Class

[TCustomMyConnection](#)

### Syntax

```
property ClientVersion: string;
```

### Remarks

Contains the version of the MySQL Client library (libmysql.dll or libmysqld.dll).

## See Also

- [TCustomDAConnection.Connect](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to specify the amount of time to attempt to establish a connection.

## Class

[TCustomMyConnection](#)

## Syntax

```
property ConnectionTimeout: integer default 15;
```

## Remarks

Use the ConnectionTimeout property to specify the amount of time to attempt to establish a connection. Use ConnectionTimeout property to specify the amount of time, in seconds, that can be expired before an attempt to make a connection is considered unsuccessful. The default value is 15 seconds.

## See Also

- [TCustomDAConnection.Connect](#)
  - [TCustomMyDataSet.CommandTimeout](#)
  - [TMyCommand.CommandTimeout](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to specify a database name that is a default source of data for SQL queries once a connection is established.

## Class

[TCustomMyConnection](#)

## Syntax

```
property Database: string;
```

## Remarks

Use the Database property to specify a database name that is a default source of data for SQL queries once a connection is established.

Altering Database property makes new database name take effect immediately.

Setting Database='mysql' allows you to omit database specifier in SELECT statements. That is, instead of

```
SELECT * FROM mysql.user;
```

you may just write

```
SELECT * FROM user
```

## See Also

- [TCustomDAConnection.Server](#)
  - [TMyConnection.Port](#)
  - [TCustomDAConnection.Username](#)
  - [TCustomDAConnection.Password](#)
  - [TCustomDAConnection.GetDatabaseNames](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to specify extent to which all outside transactions interfere with subsequent transactions of current connection.

## Class

## [TCustomMyConnection](#)

### Syntax

**property** IsolationLevel: [TMyIsolationLevel](#) **default**  
ilReadCommitted;

### Remarks

Use the IsolationLevel property to specify the extent to which all outside transactions interfere with subsequent transactions of current connection.  
Changes to IsolationLevel take effect at a time of starting new transaction or opening new connection.

### See Also

- [TCustomDAConnection.StartTransaction](#)

---

© 1997-2012 Devart. All Rights Reserved.

Specifies the behaviour of the TMyConnectionOptions object.

### Class

## [TCustomMyConnection](#)

### Syntax

**property** Options: [TCustomMyConnectionOptions](#);

### Remarks

Set the properties of Options to specify the behaviour of a TMyConnectionOptions object.  
Descriptions of all options are in the table below.

| Option Name                           | Description  |
|---------------------------------------|--|
| <a href="#">Charset</a>               | Used to set a character set used by the client.  |
| <a href="#">NullForZeroDelphiDate</a> | Used to hide the '30-12-1899' dates.   |
| <a href="#">NumericType</a>           | Used to specify the format of storing and representation of the NUMERIC (DECIMAL) fields for all <a href="#">TCustomMyDataSets</a> , associated with the given connection. |
| <a href="#">OptimisedBigInt</a>       | Used to convert all fields with field length less than 11 of TLargeIntField type into TIntegerField.   |
| <a href="#">UseUnicode</a>            | Used to inform server that all data between client and server sides will be passed in Utf8 coding.   |

### See Also

- [TCustomDAConnection.Server](#)
- [Database](#)
- [National Characters](#)

---

©

1997-2012 Devart. All Rights Reserved.

Holds the version of MySQL server.

Class

[TCustomMyConnection](#)

Syntax

**property** ServerVersion: **string**;

Remarks

The version of MySQL server.

See Also

- [TCustomDAConnection.Connect](#)

© 1997-2012 Devart. All Rights Reserved.

Used to return the thread ID of the current connection.

Class

[TCustomMyConnection](#)

Syntax

**property** ThreadId: **longword**;

Remarks

Use the ThreadId property to return the thread ID of the current connection. This value can be used as an argument to [TMyServerControl.KillProcess](#) to kill the thread.

See Also

- [TMyServerControl.KillProcess](#)

© 1997-2012 Devart. All Rights Reserved.

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

Public

| Name  | Description   |
|---|---|
| <a href="#">ApplyUpdates</a> (inherited from <a href="#">TCustomDAConnection</a> )  | Overloaded. Applies changes in datasets.  |
| <a href="#">AssignConnect</a>   | Shares database connection between the TCustomMyConnection components.                          |
| <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 customi ing line breaks in string fields and parameters.                                 |
| <a href="#">CreateDataSet</a>   | Returns a new instance of TCustomMyDataSet class and associates it with this connection object. |



|  |   |
|--|---|
| <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="#">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>  | Executes any SQL statement outside <a href="#">TMyQuery</a> or <a href="#">TMyCommand</a> components.   |
| <a href="#">ExecSQLEx</a> (inherited from <a href="#">TCustomDAConnection</a> )          | Executes any SQL statement outside the TQuery or TSQL components.                                       |
| <a href="#">GetCharsetNames</a>  | Populates a string list with the names of available charsets.   |
| <a href="#">GetDatabaseNames</a> (inherited from <a href="#">TCustomDAConnection</a> )   | Returns a database list from the server.  |
| <a href="#">GetExecuteInfo</a>   | Returns the result of the last query execution.   |
| <a href="#">GetStoredProcNames</a> (inherited from <a href="#">TCustomDAConnection</a> ) | Returns a list of stored procedures from the server.  |
| <a href="#">GetTriggerNames</a>  | Returns a list of triggers from the server.   |
| <a href="#">InTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )      | Indicates whether the transaction is active.  |
| <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="#">Options</a> (inherited from <a href="#">TCustomDAConnection</a> )            | Specifies the connection behavior.  |
| <a href="#">Password</a> (inherited from <a href="#">TCustomDAConnection</a> )           | Serves to supply a password for login.  |
| <a href="#">Ping</a>   | Allows to avoid automatic disconnection of the client by the server.                                    |
| <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="#">ReleaseSavepoint</a>   | Releases the specified savepoint without affecting any work that has been performed after its creation. |
| <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="#">RollbackToSavepoint</a>  | Cancels all updates for the current transaction.  |
| <a href="#">Savepoint</a>  | Defines a point in the transaction to which you can roll back later.                                    |

[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

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

© 1997-2012 Devart. All Rights Reserved.

Shares database connection between the TCustomMyConnection components.

### Class

[TCustomMyConnection](#)

### Syntax

```
procedure AssignConnect (Source: TCustomMyConnection); virtual;
```

#### Parameters

##### Source

Preconnected TCustomMyConnection component which connection is to be shared with the current TCustomMyConnection component.

### Remarks

Use the AssignConnect method to share database connection between the TCustomMyConnection components.

AssignConnect assumes that the Source parameter points to a preconnected TCustomMyConnection component which connection is to be shared with the current TCustomMyConnection component. Note that AssignConnect doesn't make any references to the Source TCustomMyConnection component. So before disconnecting parent TCustomMyConnection component call AssignConnect(nil) or the Disconnect method for all assigned connections.

© 1997-2012 Devart. All Rights Reserved.

Returns a new instance of TCustomMyDataSet class and associates it with this connection object.

### Class

[TCustomMyConnection](#)

### Syntax

```
function CreateDataSet: TCustomDADataSet; override;
```

#### Return Value

a new instance of TCustomMyDataSet class.

### Remarks

The CreateDataSet method returns a new instance of TCustomMyDataSet class and associates it with this connection object.

### See Also

- [CreateCommand](#)

© 1997-2012 Devart. All Rights Reserved.

Executes any SQL statement outside [TMyQuery](#) or [TMyCommand](#) components.

## Class

[TCustomMyConnection](#)

## Syntax

```
function ExecSQL(Text: string; const Params: array of variant):  
variant; override;
```

### Parameters

#### Text

Holds the SQL statement.

#### Params

Holds the array of the parameters values arranged in the same order as they appear in the SQL statement.

### Return Value

Null.

## Remarks

Call the ExecSQL method to execute any SQL statement outside [TMyQuery](#) or [TMyCommand](#) components. Supply the Params array with values of parameters arranged in the same order as they appear in SQL statement that is passed in Text string parameter.

**Note:** If a query doesn't have parameters (Params.Count = 0), this query will be executed faster.

---

© 1997-2012 Devart. All Rights Reserved.

Populates a string list with the names of available charsets.

## Class

[TCustomMyConnection](#)

## Syntax

```
procedure GetCharsetNames(List: _TStrings);
```

### Parameters

#### List

Holds the string list to populate.

## Remarks

Call the GetCharsetNames method to populate a string list with the names of available charsets.

**Note:** Any contents already in the target string list object are eliminated and overwritten by the data produced by GetCharsetNames.

## See Also

- [Options](#)

---

© 1997-2012 Devart. All Rights Reserved.

Returns the result of the last query execution.

## Class

[TCustomMyConnection](#)

## Syntax

```
function GetExecuteInfo: string;
```

### Return Value

the result of the last query execution.

## Remarks

Call the GetExecuteInfo method to returns the result of the last query execution.

The description of the result format you can see at MySQL Reference Manual [mysql info](#)

**Note:** If you execute a query at [TCustomDADataset.Execute](#) with [TCustomMyDataSet.FetchAll](#) set to False, a result cannot be retrieved.

## Example

The method makes sense for the following SQL statements:

```
INSERT INTO ... SELECT ...  
INSERT INTO ... VALUES (...), (...), (...) ...  
LOAD DATA INFILE ...  
ALTER TABLE  
UPDATE
```

## See Also

- [TCustomDADataset.Execute](#)
  - [TCustomDASQL.Execute](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Returns a list of triggers from the server.

## Class

[TCustomMyConnection](#)

## Syntax

```
procedure GetTriggerNames(List: _TStrings; AllTriggers: boolean =  
False);
```

### Parameters

#### List

A TStrings descendant that will be filled with the names of triggers in the database.

#### AllTriggers

True, if triggers from all databases are returned. False is only for the current database.

## Remarks

Call the GetTriggerNames method to get the names of triggers. GetTriggerNames populates a string list with the names of triggers in the database. If AllProcs = True, the procedure returns to the List parameter the names of the triggers that belong to all databases; otherwise, List will contain the names of triggers that belong to the current database.

**Note:** Any contents already in the target string list object are eliminated and overwritten by data produced by GetTriggerNames.

## See Also

- [TCustomDAConnection.GetDatabaseNames](#)
  - M:Devart.Dac.TCustomDAConnection.GetTableNames(Borland.Vcl.TStrings,System.Boolean)
  - [TCustomDAConnection.GetStoredProcNames](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Allows to avoid automatic disconnection of the client by the server.

### Class

[TCustomMyConnection](#)

### Syntax

```
procedure Ping;
```

### Remarks

Call the Ping method if your application has a long time intervals between accessing the server. Ping allows to avoid automatic disconnection of the client by the server. You can read the details at MySQL Reference Manual (mysql ping and wait timeout)

---

© 1997-2012 Devart. All Rights Reserved.

Releases the specified savepoint without affecting any work that has been performed after its creation.

### Class

[TCustomMyConnection](#)

### Syntax

```
procedure ReleaseSavepoint(const Name: string);
```

#### Parameters

*Name*

Holds the savepoint name.

### Remarks

Call the ReleaseSavepoint method to release the specified savepoint without affecting any work that has been performed after its creation.

### See Also

- [RollbackToSavepoint](#)
- [Savepoint](#)

---

© 1997-2012 Devart. All Rights Reserved.

Cancels all updates for the current transaction.

### Class

[TCustomMyConnection](#)

### Syntax

```
procedure RollbackToSavepoint(const Name: string);
```

#### Parameters

*Name*

Holds the name identifying the last defined savepoint.

### Remarks

Call the RollbackToSavepoint method to cancel all updates for the current transaction and restore its state up to the moment of the last defined savepoint.

### See Also

- [Savepoint](#)
  - [TCustomDAConnection.Rollback](#)
  - [ReleaseSavepoint](#)
-

© 1997-2012 Devart. All Rights Reserved.

Defines a point in the transaction to which you can roll back later.

## Class

[TCustomMyConnection](#)

## Syntax

```
procedure Savepoint(const Name: string);
```

### Parameters

*Name*

Holds the name of the savepoint.

## Remarks

Call the Savepoint 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 [RollbackToSavepoint](#).

## See Also

- [RollbackToSavepoint](#)
- [ReleaseSavepoint](#)

© 1997-2012 Devart. All Rights Reserved.

### 17.15.1.2 MyAccess.TCustomMyConnectionOptions Class

This class allows setting up the behaviour of the TCustomMyConnection class.

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

## Unit

[MyAccess](#)

## Syntax

```
TCustomMyConnectionOptions = class (TDAConnectionOptions);
```

## Inheritance Hierarchy

TObject

[TDAConnectionOptions](#)

**TCustomMyConnectionOptions**

© 1997-2012 Devart. All Rights Reserved.

[TCustomMyConnectionOptions](#) class overview.

## Properties

| Name  | Description   |
|---|---|
| <a href="#">Charset</a>   | Used to set a character set used by the client.   |
| <a href="#">DefaultSortType</a> (inherited from <a href="#">TDAConnectionOptions</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="#">TDAConnectionOptions</a> ) | Used to open a connection only when needed for performing a server call and closes after performing the operation.  |

|  |  |
|--|--|
| <a href="#">KeepDesignConnected</a> (inherited from <a href="#">TDAConnectionOptions</a> ) | Used to prevent an application from establishing a connection at the time of startup.  |
| <a href="#">LocalFailover</a> (inherited from <a href="#">TDAConnectionOptions</a> )       | If True, the <a href="#">TCustomDAConnection.OnConnectionLost</a> event occurs and a failover operation can be performed after connection breaks.                          |
| <a href="#">NullForZeroDelphiDate</a>  | Used to hide the '30-12-1899' dates.   |
| <a href="#">NumericType</a>  | Used to specify the format of storing and representation of the NUMERIC (DECIMAL) fields for all <a href="#">TCustomMyDataSet</a> s, associated with the given connection. |
| <a href="#">OptimisedBigInt</a>  | Used to convert all fields with field length less than 11 of TLargeIntField type into TIntegerField.   |
| <a href="#">UseUnicode</a>   | Used to inform server that all data between client and server sides will be passed in Utf8 coding.   |

© 1997-2012 Devart. All Rights Reserved.

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

## Public

| Name   | Description   |
|--|---|
| <a href="#">Charset</a>  | Used to set a character set used by the client.   |
| <a href="#">DefaultSortType</a> (inherited from <a href="#">TDAConnectionOptions</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="#">TDAConnectionOptions</a> )    | Used to open a connection only when needed for performing a server call and closes after performing the operation.  |
| <a href="#">KeepDesignConnected</a> (inherited from <a href="#">TDAConnectionOptions</a> ) | Used to prevent an application from establishing a connection at the time of startup.   |
| <a href="#">LocalFailover</a> (inherited from <a href="#">TDAConnectionOptions</a> )       | If True, the <a href="#">TCustomDAConnection.OnConnectionLost</a> event occurs and a failover operation can be performed after connection breaks.   |
| <a href="#">NullForZeroDelphiDate</a>  | Used to hide the '30-12-1899' dates.  |
| <a href="#">NumericType</a>  | Used to specify the format of storing and representation of the NUMERIC (DECIMAL) fields for all <a href="#">TCustomMyDataSet</a> s, associated with the given connection.  |
| <a href="#">OptimisedBigInt</a>  | Used to convert all fields with field length less than 11 of TLargeIntField type into TIntegerField.  |

[UseUnicode](#)

Used to inform server that all data between client and server sides will be passed in Utf8 coding.

### See Also

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

© 1997-2012 Devart. All Rights Reserved.

Used to set a character set used by the client.

### Class

[TCustomMyConnectionOptions](#)

### Syntax

```
property Charset: string;
```

### Remarks

Use the Charset property to set a character set used by the client. Actually, if this property is enabled, then the "SET NAMES <Charset>" query is executed on establishing a connection. If the UseUnicode property is set, the Charset property will be ignored. The list of available character sets you can see by executing the [TCustomMyConnection.GetCharsetNames](#) method.

---

© 1997-2012 Devart. All Rights Reserved.

Used to hide the '30-12-1899' dates.

### Class

[TCustomMyConnectionOptions](#)

### Syntax

```
property NullForZeroDelphiDate: boolean default False;
```

### Remarks

Use the NullForZeroDelphiDate property to hide the '30-12-1899' dates. If NullForZeroDelphiDate is set to True, the values of all datetime fields will be changed to Null. If the property is set to False, the '30-12-1899' value will be used as an ordinary date. The default value is False.

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify the format of storing and representation of the NUMERIC (DECIMAL) fields for all [TCustomMyDataSets](#), associated with the given connection.

### Class

[TCustomMyConnectionOptions](#)

### Syntax

```
property NumericType: TDANumericType default ntFloat;
```

### Remarks

Use the NumericType property to specify the format of storing and representation of the NUMERIC (DECIMAL) fields for all [TCustomMyDataSets](#), associated with the given connection.

---

© 1997-2012 Devart. All Rights Reserved.



Used to convert all fields with field length less than 11 of TLargeIntField type into TIntegerField.

### Class

[TCustomMyConnectionOptions](#)

### Syntax

```
property OptimizedBigInt: boolean default False;
```

### Remarks

Setting this option converts all fields with field length less than 11 of TLargeIntField type into TIntegerField. This allows to process fields that are results of numeric function or cast values as usual Integer fields. The default value is False.

© 1997-2012 Devart. All Rights Reserved.

Used to inform server that all data between client and server sides will be passed in Utf8 coding.

### Class

[TCustomMyConnectionOptions](#)

### Syntax

```
property UseUnicode: boolean default False;
```

### Remarks

Informs server that all data between client and server sides will be passed in Utf8 coding. Setting this option converts all fields of the TStringField type into TWideStringField that allows to work correctly with symbols of almost all languages simultaneously. On the other hand, it causes a delay in working. If the UseUnicode property is enabled, the Charset property will be ignored. The default value is False.

© 1997-2012 Devart. All Rights Reserved.

#### 17.15.1.3 MyAccess.TCustomMyDataSet Class

A base class defining functionality for the classes derived from it.

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

### Unit

[MyAccess](#)

### Syntax

```
TCustomMyDataSet = class (TCustomDADataset);
```

### Remarks

TCustomMyDataSet is a base dataset component that defines functionality for classes derived from it. Applications never use TCustomMyDataSet objects directly. Instead they use descendants of TCustomMyDataSet, such as TMyQuery and TMyTable that inherit its dataset-related properties and methods.

### Inheritance Hierarchy

```

TObject
  TMemDataSet
    TCustomDADataset
      TCustomMyDataSet

```

### See Also

- [TMyQuery](#)
- [TCustomMyTable](#)
- [Master/Detail Relationships](#)

© 1997-2012 Devart. All Rights Reserved.

[TCustomMyDataSet](#) class overview.

## Properties

| Name  | Description  |
|---|--|
| <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="#">CommandTimeout</a>  | Used to specify the amount of time to attempt execution of a command.  |
| <a href="#">Connection</a>  | Used to specify a connection object that will be used to connect to a data store.  |
| <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="#">Encryption</a> (inherited from <a href="#">TCustomDADataset</a> )   | Used to specify the options of the data encryption in a dataset.   |
| <a href="#">FetchAll</a>  | Description is not available at the moment.  |
| <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="#">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="#">InsertId</a>  | Returns the ID generated for an AUTO INCREMENT column by the previous query.   |
| <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="#">LockMode</a>  | Specifies when to perform locking of an 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>   | Specifies the behaviour of TCustomMyDataSet 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="#">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="#">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="#">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="#">UpdateRecordTypes</a> (inherited from <a href="#">TMemDataSet</a> )   | 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.

## Methods

### Name

[AddWhere](#) (inherited from [TCustomDADataset](#))

[ApplyUpdates](#) (inherited from [TMemDataSet](#))

[CancelUpdates](#) (inherited from [TMemDataSet](#))

[CommitUpdates](#) (inherited from [TMemDataSet](#))

[CreateBlobStream](#) (inherited from [TCustomDADataset](#))

[DeferredPost](#) (inherited from [TMemDataSet](#))

[DeleteWhere](#) (inherited from [TCustomDADataset](#))

[Execute](#) (inherited from [TCustomDADataset](#))

[Executing](#) (inherited from [TCustomDADataset](#))

[Fetched](#) (inherited from [TCustomDADataset](#))

[Fetching](#) (inherited from [TCustomDADataset](#))

[FetchingAll](#) (inherited from [TCustomDADataset](#))

[FindKey](#) (inherited from [TCustomDADataset](#))

[FindMacro](#) (inherited from [TCustomDADataset](#))

[FindNearest](#) (inherited from [TCustomDADataset](#))

[FindParam](#) (inherited from [TCustomDADataset](#))

[GetBlob](#) (inherited from [TMemDataSet](#))

[GetDataType](#) (inherited from [TCustomDADataset](#))

[GetFieldEnum](#)

[GetFieldObject](#) (inherited from [TCustomDADataset](#))

### Description

Adds condition to the WHERE clause of SELECT statement in the SQL property.

Overloaded. Writes dataset's pending cached updates to a database.

Clears all pending cached updates from cache and restores dataset in its prior state.

Clears the cached updates buffer.

Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.

Makes permanent changes to the database server.

Removes WHERE clause from the SQL property and assigns the BaseSQL property.

Executes a SQL statement on the server.

Indicates whether SQL statement is still being executed.

Used to learn whether TCustomDADataset has already fetched all rows.

Used to learn whether TCustomDADataset is still fetching rows.

Used to learn whether TCustomDADataset is fetching all rows to the end.

Searches for a record which contains specified field values.

Indicates whether a specified macro exists in a dataset.

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.

Determines if a parameter with the specified name exists in a dataset.

Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.

Returns internal field types defined in the MemData and accompanying modules.

Retrieve the list of acceptable values for a specified field given by the FieldName parameter.

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>   | Overloaded. Locks the current record for the current connection.   |
| <a href="#">LockTable</a>  | Locks table for the current connection.  |
| <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="#">TCustomDADataset</a> )       | Sets or uses 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="#">RefreshQuick</a>   | Retrieves changes posted to the server by another clients on the client side quickly.                                      |
| <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="#">UnLockTable</a>  | Releases a table locked by the <a href="#">TCustomMyDataSet.LockTable</a> method.  |

[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.                     |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TCustomMyDataSet** class.

For a complete list of the **TCustomMyDataSet** class members, see the [TCustomMyDataSet 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="#"><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>CommandTimeout</u></a>  | Used to specify the amount of time to attempt execution of a command.  |
| <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>  | Used to specify a connection object that will be used to connect to a data store.  |
| <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>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>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>FetchAll</u></a>  | Description is not available at the moment.  |
| <a href="#"><u>Fetched</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.  |

[FinalSQL](#) (inherited from [TCustomDADataset](#))

[FindKey](#) (inherited from [TCustomDADataset](#))

[FindMacro](#) (inherited from [TCustomDADataset](#))

[FindNearest](#) (inherited from [TCustomDADataset](#))

[FindParam](#) (inherited from [TCustomDADataset](#))

[GetBlob](#) (inherited from [TMemDataSet](#))

[GetDataType](#) (inherited from [TCustomDADataset](#))

[GetFieldObject](#) (inherited from [TCustomDADataset](#))

[GetFieldPrecision](#) (inherited from [TCustomDADataset](#))

[GetFieldScale](#) (inherited from [TCustomDADataset](#))

[GetOrderBy](#) (inherited from [TCustomDADataset](#))

[GotoCurrent](#) (inherited from [TCustomDADataset](#))

[IndexFieldNames](#) (inherited from [TMemDataSet](#))

[InsertId](#)

[IsQuery](#) (inherited from [TCustomDADataset](#))

[KeyFields](#) (inherited from [TCustomDADataset](#))

[LocalConstraints](#) (inherited from [TMemDataSet](#))

[LocalUpdate](#) (inherited from [TMemDataSet](#))

[Locate](#) (inherited from [TMemDataSet](#))

[LocateEx](#) (inherited from [TMemDataSet](#))

[Lock](#) (inherited from [TCustomDADataset](#))

Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.

Searches for a record which contains specified field values.

Indicates whether a specified macro exists in a dataset.

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.

Determines if a parameter with the specified name exists in a dataset.

Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.

Returns internal field types defined in the MemData and accompanying modules.

Returns a multireference shared object from field.

Retrieves the precision of a number field.

Retrieves the scale of a number field.

Retrieves an ORDER BY clause from a SQL statement.

Sets the current record in this dataset similar to the current record in another dataset.

Used to get or set the list of fields on which the recordset is sorted.

Returns the ID generated for an AUTO INCREMENT column by the previous query.

Used to check whether SQL statement returns rows.

Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.

Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

Used to prevent implicit update of rows on database server.

Overloaded. Searches a dataset for a specific record and positions the cursor on it.

Overloaded. Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet.

Locks the current record.



|   |  |
|---|--|
| <a href="#">LockMode</a>  | Specifies when to perform locking of an 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>   | Specifies the behaviour of TCustomMyDataSet 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="#">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.  |

[RevertRecord](#) (inherited from [TMemDataSet](#))

[RowsAffected](#) (inherited from [TCustomDADataset](#))

[SaveSQL](#) (inherited from [TCustomDADataset](#))

[SaveToXML](#) (inherited from [TMemDataSet](#))

[SetOrderBy](#) (inherited from [TCustomDADataset](#))

[SQL](#) (inherited from [TCustomDADataset](#))

[SQLDelete](#) (inherited from [TCustomDADataset](#))

[SQLInsert](#) (inherited from [TCustomDADataset](#))

[SQLLock](#) (inherited from [TCustomDADataset](#))

[SQLRefresh](#) (inherited from [TCustomDADataset](#))

[SQLSaved](#) (inherited from [TCustomDADataset](#))

[SQLUpdate](#) (inherited from [TCustomDADataset](#))

[UniDirectional](#) (inherited from [TCustomDADataset](#))

[UnLock](#) (inherited from [TCustomDADataset](#))

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

[UpdatesPending](#) (inherited from [TMemDataSet](#))

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Cancels changes made to the current record when cached updates are enabled.

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

Saves the SQL property value to BaseSQL.

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Builds an ORDER BY clause of a SELECT statement.

Used to provide a SQL statement that a query component executes when its Open method is called.

Used to specify a SQL statement that will be used when applying a deletion to a record.

Used to specify the SQL statement that will be used when applying an insertion to a dataset.

Used to specify a SQL statement that will be used to perform a record lock.

Used to specify a SQL statement that will be used to refresh current record by calling the [TCustomDADataset.RefreshRecord](#) procedure.

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

Used to specify a SQL statement that will be used when applying an update to a dataset.

Used if an application does not need bidirectional access to records in the result set.

Releases a record lock.

Frees the resources allocated for a previously prepared query on the server and client sides.

Used to indicate the update status for the current record when cached updates are enabled.

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

Used to check the status of the cached updates buffer.

Indicates the current update status for the dataset when cached updates are enabled.

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Used to specify the amount of time to attempt execution of a command.

## Class

[TCustomMyDataSet](#)

## Syntax

```
property CommandTimeout: integer default 0;
```

## Remarks

Use the CommandTimeout property to specify the amount of time to attempt execution of a command. Use CommandTimeout to specify the amount of time that expires before an attempt to execute a command is considered unsuccessful. Is measured in seconds.

If a command is successfully executed prior to the expiration of the seconds specified, CommandTimeout has no effect.

In the case of exceeding waiting time error CR\_SERVER\_LOST 'Lost connection to MySQL server during query' raises.

The default value is 0 (infinite).

## See Also

- [TCustomMyConnection.ConnectionTimeout](#)
- [TMyCommand.CommandTimeout](#)

© 1997-2012 Devart. All Rights Reserved.

Used to specify a connection object that will be used to connect to a data store.

## Class

[TCustomMyDataSet](#)

## Syntax

```
property Connection: TCustomMyConnection;
```

## 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 TCustomMyConnection objects.

At runtime, set the Connection property to reference an existing TCustomMyConnection object.

## See Also

- [TCustomMyConnection](#)

© 1997-2012 Devart. All Rights Reserved.

## Class

[TCustomMyDataSet](#)

## Syntax

```
property FetchAll: boolean default True;
```

## Remarks

The default value is True.

**Note:** When setting [TCustomMyDataSet.FetchAll](#) = False you should keep in mind that execution of such queries blocks current session. By default MyDAC will create additional session if it is necessary. But this can cause the following problems:

- Each additional session runs outside the transaction context thus TMyConnection.

[TCustomDAConnection.Commit](#) and TMyConnection.[TCustomDAConnection.Rollback](#) operations in main session won't apply changes made in additional sessions.

- Temporary tables created in one session are not accessible from other sessions, therefore simultaneous using of FetchAll = False and temporary tables is impossible.
- [TCustomMyDataSet.Lock](#) cannot be used
- [LockTable](#) cannot be used

In order to avoid creating additional connection you can set [Options](#) to False.

© 1997-2012 Devart. All Rights Reserved.

Returns the ID generated for an AUTO INCREMENT column by the previous query.

## Class

[TCustomMyDataSet](#)

## Syntax

**property** InsertId: int64;

## Remarks

Use the InsertId property to return the ID generated for an AUTO INCREMENT column by the previous query. Use this function after you have performed an INSERT query into a table that contains an AUTO INCREMENT field.

If the query does not perform an insertion into a table that contains an AUTO INCREMENT field the value of InsertId won't be defined.

InsertId property has sense only if SQL includes INSERT statement within itself. In case of SELECT statements, the value of auto-increment field can be obtained from corresponding table fields.

## See Also

- [TCustomDADataSet.Execute](#)

© 1997-2012 Devart. All Rights Reserved.

Specifies when to perform locking of an editing record.

## Class

[TCustomMyDataSet](#)

## Syntax

**property** LockMode: [TLockMode](#);

## 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.

© 1997-2012 Devart. All Rights Reserved.

Specifies the behaviour of TCustomMyDataSet object.

## Class

[TCustomMyDataSet](#)

## Syntax

**property** Options: [TMyDataSetOptions](#);

## Remarks

Set the properties of Options to specify the behaviour of a TCustomMyDataSet object. Descriptions of all options are in the table below.

| Option Name | Description |
|-------------|-------------|
|-------------|-------------|

|                                     |  |
|-------------------------------------|--|
| <a href="#">AutoPrepare</a>         | Used to execute automatic <a href="#">TCustomDADataset.Prepare</a> on a query execution.   |
| <a href="#">AutoRefresh</a>         | Used to automatically refresh dataset every AutoRefreshInterval seconds.   |
| <a href="#">AutoRefreshInterval</a> | Used to define in what time interval in seconds the Refresh or <a href="#">RefreshQuick</a> method of a DataSet is called.   |
| <a href="#">BinaryAsString</a>      | Used to specify a method of representation of the BINARY and VARBINARY fields.   |
| <a href="#">CheckRowVersion</a>     | Used to determine whether a dataset checks for rows modifications made by another user on automatic generation of SQL statement for update or delete data.                     |
| <a href="#">CreateConnection</a>    | Used to specify if an additional connection to a server should be established to execute an additional query in the <a href="#">FetchAll=False</a> mode.                       |
| <a href="#">DefaultValues</a>       | Used to fill the DefaultExpression property of TField objects with appropriate value.  |
| <a href="#">EnableBoolean</a>       | Used to specify the method of representation of the TINYINT(1) fields.   |
| <a href="#">FieldsAsString</a>      | Used to store all non-BLOB fields as string (native MySQL format).   |
| <a href="#">FieldsOrigin</a>        | Used to fill the Origin property of TField objects with appropriate value.   |
| <a href="#">FullRefresh</a>         | Used to specify the fields to include in automatically generated SQL statement when calling the <a href="#">TCustomDADataset.RefreshRecord</a> method. Default value is false. |
| <a href="#">NullForZeroDate</a>     | Used for MySQL server to represent the value for for datetime fields with invalid values as Null or '0001-01-01' ('0100-01-01' for CLR).                                       |
| <a href="#">NumberRange</a>         | Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.   |
| <a href="#">QueryRecCount</a>       | Used for TCustomDADataset to perform additional query to get record count for this SELECT so the RecordCount property reflects the actual number of records.                   |
| <a href="#">QuoteNames</a>          | Used for TCustomMyDataSet to quote all field names in autogenerated SQL statements.  |
| <a href="#">RemoveOnRefresh</a>     | Used for dataset to remove a record locally if the RefreshRecord procedure can't find necessary record on the server.  |
| <a href="#">RequiredFields</a>      | Used for TCustomDADataset to set the Required property of TField objects for NOT NULL fields.  |
| <a href="#">ReturnParams</a>        | Used to return the new value of the fields to dataset after insert or update.  |
| <a href="#">SetFieldsReadOnly</a>   | Used to specify whether fields not belonging to the current updating table get read-only attribute.  |
| <a href="#">StrictUpdate</a>        | Used for TCustomDADataset to raise an exception when the number of updated or deleted records does not equal 1.  |
| <a href="#">TrimFixedChar</a>       | Used to specify whether to discard all trailing spaces in string fields of the dataset.  |

1997-2012 Devart. All Rights Reserved.

Methods of the **TCustomMyDataSet** class.

For a complete list of the **TCustomMyDataSet** class members, see the [TCustomMyDataSet 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="#">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="#"><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="#"><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>TCustomDADataset</u></a> )         | Determines if a parameter with the specified name exists in a dataset.   |
| <a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )                | Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself 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>GetFieldEnum</u></a>  | Retrieve the list of acceptable values for a specified field given by the FieldName parameter.   |
| <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>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>TCustomDADataset</u></a> )           | Used to check whether 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>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>   | Overloaded. Locks the current record for the current connection.   |
| <a href="#"><u>LockTable</u></a>  | Locks table for the current connection.  |
| <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> (inherited from <a href="#"><u>TCustomDADataset</u></a> )  | Sets or uses 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>Prepare</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></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="#">RefreshQuick</a>  | Retrieves changes posted to the server by another clients on the client side quickly.  |
| <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.  |

[UnLock](#) (inherited from [TCustomDADDataSet](#))  
[UnLockTable](#)

Releases a record lock.  
 Releases a table locked by the [TCustomMyDataSet.LockTable](#) method.

[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

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

© 1997-2012 Devart. All Rights Reserved.

Retrieve the list of acceptable values for a specified field given by the FieldName parameter.

### Class

[TCustomMyDataSet](#)

### Syntax

```
procedure GetFieldEnum(List: _TStrings; FieldName: string;
  TableName: string = '');;
```

#### Parameters

##### List

holds the list of acceptable values for a specified field.

##### FieldName

Holds the field name.

##### TableName

Holds the table name.

### Remarks

Call the GetFieldEnum method to retrieve the list of acceptable values for a specified field given by the FieldName parameter. Field should be of the ENUM or the SET type. If GetFieldEnum is called with the TableName parameter empty, TCustomMyDataSet tries to determine table name searching corresponding field name in the fields requested from server.

### Example

The code presented in Example 1) demonstrates the usage of the GetFieldEnum method. This code can be tested on the sample table presented in Example 2). The result output in memo is shown in Example 3).

```
Example 1)
MyQuery.SQL.Text := 'SELECT `id`, `SET_column` FROM tb_with_set_column';
MyQuery.Open;
MyQuery.GetFieldEnum(Memo.Lines, 'SET_column');
Example 2)
```

```

DROP TABLE if EXISTS tb_enum;
CREATE TABLE `tb_enum` (
  `uid` INT(11) not NULL PRIMARY KEY AUTO_INCREMENT,
  `c_enum` ENUM('value1','value2','value3') DEFAULT NULL
);
Example 3)
value1
value2
value3

```

© 1997-2012 Devart. All Rights Reserved.

Locks the current record for the current connection.

## Class

[TCustomMyDataSet](#)

## Overload List

| Name  | Description  |
|---|--|
| <a href="#">Lock</a>                            | Locks the current record for the current connection. |
| <a href="#">Lock(LockType: TLockRecordType)</a> | Locks the current record for the current connection. |

© 1997-2012 Devart. All Rights Reserved.

Locks the current record for the current connection.

## Class

[TCustomMyDataSet](#)

## Syntax

**procedure** Lock; **overload;** **override**

## Remarks

Locks the current record for the current connection. Serves to prevent simultaneous editing of the same record by several users. Makes sense only for InnoDB tables and can be called only inside transaction context.

If any other connection tries to modify the locked record, it will wait for the time specified by innodb lock wait timeout server variable, by default 50 seconds. If during this time the record will not be unlocked, an exception raises.

The record is unlocked on finishing transaction (Commit or Rollback).

**Note:** This method is incompatible with the [TCustomMyDataSet.FetchAll](#) property set to False.

Lock can be used only with queries returning resultset.

## See Also

- [TCustomDAConnection.StartTransaction](#)
- [TCustomDAConnection.Commit](#)
- [TCustomDAConnection.Rollback](#)
- [TCustomMyDataSet.LockTable](#)

© 1997-2012 Devart. All Rights Reserved.

Locks the current record for the current connection.

## Class

[TCustomMyDataSet](#)

### Syntax

```
procedure Lock (LockType: TLockRecordType); reintroduce; overload
```

**Parameters**

*LockType*  
Holds the type of the record locking.

---

© 1997-2012 Devart. All Rights Reserved.

Locks table for the current connection.

### Class

[TCustomMyDataSet](#)

### Syntax

```
procedure LockTable (LockType: TLockType);
```

**Parameters**

*LockType*  
Specifies the type of the table locking.

### Remarks

Call the LockTable method to lock table for the current connection. The main purpose of this method is to speed up working with the table.

Table can be released on:

- calling UnlockTable;
- closing connection;
- calling LockTable once more for the same connection.

If a query has several tables then a table specified in [TMyQuery.UpdatingTable](#) is used.

**Note:** This method is incompatible with the [FetchAll](#) property set to False.

### See Also

- [UnlockTable](#)
  - [TCustomMyDataSet.Lock](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Retrieves changes posted to the server by another clients on the client side quickly.

### Class

[TCustomMyDataSet](#)

### Syntax

```
procedure RefreshQuick (const CheckDeleted: boolean);
```

**Parameters**

*CheckDeleted*  
True, if records deleted by another clients will be checked additionally.

### Remarks

Call the RefreshQuick method to retrieve changes posted to the server by another clients on the client side quickly. The main difference from the Refresh method is that not all data corresponding to the query is retrieved on the client, but only the rows which were added or modified at the moment of the last update. A necessity of data inquiry for each row is defined by TIMESTAMP field.

If CheckDeleted parameter set to True records deleted by another clients will be checked additionally. For RefreshQuick to work it is necessary that a query includes unique key fields and TIMESTAMP field. This method is especially effective for queries with huge data level in the single row.

**Note:** If RefreshQuick is called for a dataset which is ordered on the server (query includes the ORDER BY clause), dataset records ordering can be violated because not all records will be retrieved by this method. You can use local ordering to solve this problem. For more information about local ordering, see

the [TMemDataSet.IndexFieldNames](#) property description.

© 1997-2012 Devart. All Rights Reserved.

Releases a table locked by the [LockTable](#) method.

## Class

[TCustomMyDataSet](#)

## Syntax

```
procedure UnLockTable;
```

## Remarks

Call the UnLockTable method to release a table locked by [LockTable](#).

## See Also

- [LockTable](#)
- [TCustomMyDataSet.Lock](#)

© 1997-2012 Devart. All Rights Reserved.

### 17.15.1.4 MyAccess.TCustomMyStoredProc Class

A class implementing functionality to access stored procedures on a database server.

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

## Unit

[MyAccess](#)

## Syntax

```
TCustomMyStoredProc = class(TCustomMyDataSet);
```

## Remarks

TCustomMyStoredProc implements functionality to access stored procedures on a database server. You need only to define the StoredProcName property, while not bothering about writing a SQL statement manually.

Use the Execute method at runtime to generate a request that instructs server to execute procedure and return parameters in the Params property.

Stored procedures are supported only for MySQL 5.0.

## Inheritance Hierarchy

```

TObject
  TMemDataSet
    TCustomDADataset
      TCustomMyDataSet
        TCustomMyStoredProc

```

## See Also

- [TCustomMyDataSet](#)
- [TMyStoredProc](#)

© 1997-2012 Devart. All Rights Reserved.

[TCustomMyStoredProc](#) class overview.

## Properties

| Name | Description |
|------|-------------|
|------|-------------|

|   |  |
|---|--|
| <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="#">CommandTimeout</a> (inherited from <a href="#">TCustomMyDataSet</a> ) | Used to specify the amount of time to attempt execution of a command.  |
| <a href="#">Connection</a> (inherited from <a href="#">TCustomMyDataSet</a> )     | Used to specify a connection object that will be used to connect to a data store.  |
| <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="#">Encryption</a> (inherited from <a href="#">TCustomDADataset</a> )     | Used to specify the options of the data encryption in a dataset.   |
| <a href="#">FetchAll</a> (inherited from <a href="#">TCustomMyDataSet</a> )       | Description is not available at the moment.  |
| <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="#">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="#">InsertId</a> (inherited from <a href="#">TCustomMyDataSet</a> )       | Returns the ID generated for an AUTO INCREMENT column by the previous query.   |
| <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="#">LockMode</a> (inherited from <a href="#">TCustomMyDataSet</a> )       | Specifies when to perform locking of an 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="#">TCustomMyDataSet</a> )        | Specifies the behaviour of TCustomMyDataSet 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="#">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="#">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="#">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="#">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.   |

[UpdatesPending](#) (inherited from [TMemDataSet](#))

Used to check the status of the cached updates buffer.

## Methods

### Name

[AddWhere](#) (inherited from [TCustomDADataset](#))

[ApplyUpdates](#) (inherited from [TMemDataSet](#))

[CancelUpdates](#) (inherited from [TMemDataSet](#))

[CommitUpdates](#) (inherited from [TMemDataSet](#))

[CreateBlobStream](#) (inherited from [TCustomDADataset](#))

[DeferredPost](#) (inherited from [TMemDataSet](#))

[DeleteWhere](#) (inherited from [TCustomDADataset](#))

[ExecProc](#)

[Execute](#) (inherited from [TCustomDADataset](#))

[Executing](#) (inherited from [TCustomDADataset](#))

[Fetched](#) (inherited from [TCustomDADataset](#))

[Fetching](#) (inherited from [TCustomDADataset](#))

[FetchingAll](#) (inherited from [TCustomDADataset](#))

[FindKey](#) (inherited from [TCustomDADataset](#))

[FindMacro](#) (inherited from [TCustomDADataset](#))

[FindNearest](#) (inherited from [TCustomDADataset](#))

[FindParam](#) (inherited from [TCustomDADataset](#))

[GetBlob](#) (inherited from [TMemDataSet](#))

[GetDataType](#) (inherited from [TCustomDADataset](#))

[GetFieldEnum](#) (inherited from [TCustomMyDataSet](#))

### Description

Adds condition to the WHERE clause of SELECT statement in the SQL property.

Overloaded. Writes dataset's pending cached updates to a database.

Clears all pending cached updates from cache and restores dataset in its prior state.

Clears the cached updates buffer.

Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.

Makes permanent changes to the database server.

Removes WHERE clause from the SQL property and assigns the BaseSQL property.

Executes a SQL statement on the server.

Executes a SQL statement on the server.

Indicates whether SQL statement is still being executed.

Used to learn whether TCustomDADataset has already fetched all rows.

Used to learn whether TCustomDADataset is still fetching rows.

Used to learn whether TCustomDADataset is fetching all rows to the end.

Searches for a record which contains specified field values.

Indicates whether a specified macro exists in a dataset.

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.

Determines if a parameter with the specified name exists in a dataset.

Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.

Returns internal field types defined in the MemData and accompanying modules.

Retrieve the list of acceptable values for a specified field given by the FieldName parameter.



|  |   |
|--|---|
| <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>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>TCustomMyDataSet</u></a> )              | Overloaded. Locks the current record for the current connection.  |
| <a href="#"><u>LockTable</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )         | Locks table for the current connection.   |
| <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>ParamByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )       | Sets or uses parameter information for a specific parameter based on its name.  |
| <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>PrepareSQL</u></a>  | Builds a query for TCustomMyStoredProc based on the Params and StoredProcName properties, and assigns it to the SQL property.     |
| <a href="#"><u>RefreshQuick</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Retrieves changes posted to the server by another clients on the client side quickly.   |
| <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="#"><u>Resync</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )            | Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.           |
| <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>SaveSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )           | Saves the SQL property value to BaseSQL.  |
| <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>SetOrderBy</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )        | Builds an ORDER BY clause of a SELECT statement.  |

[SQLSaved](#) (inherited from [TCustomDADataset](#))

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

[UnLock](#) (inherited from [TCustomDADataset](#))

Releases a record lock.

[UnLockTable](#) (inherited from [TCustomMyDataSet](#))

Releases a table locked by the [TCustomMyDataSet.LockTable](#) method.

[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.                     |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TCustomMyStoredProc** class.

For a complete list of the **TCustomMyStoredProc** class members, see the [TCustomMyStoredProc 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 <a href="#">SQL</a> 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>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>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>CommandTimeout</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Used to specify the amount of time to attempt execution of a command.  |
| <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>TCustomMyDataSet</u></a> )          | Used to specify a connection object that will be used to connect to a data store.  |
| <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>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>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>FetchAll</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )            | Description is not available at the moment.  |
| <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.   |

[FilterSQL](#) (inherited from [TCustomDADataset](#))

[FinalSQL](#) (inherited from [TCustomDADataset](#))

[FindKey](#) (inherited from [TCustomDADataset](#))

[FindMacro](#) (inherited from [TCustomDADataset](#))

[FindNearest](#) (inherited from [TCustomDADataset](#))

[FindParam](#) (inherited from [TCustomDADataset](#))

[GetBlob](#) (inherited from [TMemDataSet](#))

[GetDataType](#) (inherited from [TCustomDADataset](#))

[GetFieldEnum](#) (inherited from [TCustomMyDataSet](#))

[GetFieldObject](#) (inherited from [TCustomDADataset](#))

[GetFieldPrecision](#) (inherited from [TCustomDADataset](#))

[GetFieldScale](#) (inherited from [TCustomDADataset](#))

[GetOrderBy](#) (inherited from [TCustomDADataset](#))

[GotoCurrent](#) (inherited from [TCustomDADataset](#))

[IndexFieldNames](#) (inherited from [TMemDataSet](#))

[InsertId](#) (inherited from [TCustomMyDataSet](#))

[IsQuery](#) (inherited from [TCustomDADataset](#))

[KeyFields](#) (inherited from [TCustomDADataset](#))

[LocalConstraints](#) (inherited from [TMemDataSet](#))

[LocalUpdate](#) (inherited from [TMemDataSet](#))

[Locate](#) (inherited from [TMemDataSet](#))

Used to change the WHERE clause of SELECT statement and reopen a query.

Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.

Searches for a record which contains specified field values.

Indicates whether a specified macro exists in a dataset.

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.

Determines if a parameter with the specified name exists in a dataset. Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.

Returns internal field types defined in the MemData and accompanying modules.

Retrieve the list of acceptable values for a specified field given by the FieldName parameter.

Returns a multireference shared object from field.

Retrieves the precision of a number field.

Retrieves the scale of a number field.

Retrieves an ORDER BY clause from a SQL statement.

Sets the current record in this dataset similar to the current record in another dataset.

Used to get or set the list of fields on which the recordset is sorted.

Returns the ID generated for an AUTO INCREMENT column by the previous query.

Used to check whether SQL statement returns rows.

Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.

Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

Used to prevent implicit update of rows on database server.

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="#">TCustomMyDataSet</a> )           | Overloaded. Locks the current record for the current connection.   |
| <a href="#">LockMode</a> (inherited from <a href="#">TCustomMyDataSet</a> )       | Specifies when to perform locking of an editing record.  |
| <a href="#">LockTable</a> (inherited from <a href="#">TCustomMyDataSet</a> )      | Locks table for the current connection.  |
| <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> (inherited from <a href="#">TCustomMyDataSet</a> )        | Specifies the behaviour of TCustomMyDataSet 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="#">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="#">RefreshQuick</a> (inherited from <a href="#">TCustomMyDataSet</a> )   | Retrieves changes posted to the server by another clients on the client side quickly.  |

|   |  |
|---|--|
| <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="#">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="#">StoredProcName</a>  | Used to specify the name of the stored procedure to call on the server.  |
| <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="#">UnLockTable</a> (inherited from <a href="#">TCustomMyDataSet</a> )    | Releases a table locked by the <a href="#">TCustomMyDataSet.LockTable</a> method.  |
| <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

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

© 1997-2012 Devart. All Rights Reserved.

Used to specify the name of the stored procedure to call on the server.

## Class

[TCustomMyStoredProc](#)

## 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.

© 1997-2012 Devart. All Rights Reserved.

Methods of the **TCustomMyStoredProc** class.

For a complete list of the **TCustomMyStoredProc** class members, see the [TCustomMyStoredProc 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="#"><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>CommandTimeout</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )   | Used to specify the amount of time to attempt execution of a command.  |
| <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>TCustomMyDataSet</u></a> )       | Used to specify a connection object that will be used to connect to a data store.  |
| <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>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>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>FetchAll</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )         | Description is not available at the moment.  |
| <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="#"><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>TCustomDADataset</u></a> )         | Determines if a parameter with the specified name exists in a dataset.   |
| <a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )                | Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself 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>GetFieldEnum</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Retrieve the list of acceptable values for a specified field given by the FieldName parameter.   |
| <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>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>InsertId</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )          | Returns the ID generated for an AUTO INCREMENT column by the previous query.   |
| <a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )           | Used to check whether 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>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>TCustomMyDataSet</u></a> )              | Overloaded. Locks the current record for the current connection.   |

|   |  |
|---|--|
| <a href="#"><u>LockMode</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )       | Specifies when to perform locking of an editing record.  |
| <a href="#"><u>LockTable</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Locks table for the current connection.  |
| <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>TCustomMyDataSet</u></a> )        | Specifies the behaviour of TCustomMyDataSet object.  |
| <a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )    | Sets or uses 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>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>PrepareSQL</u></a>   | Builds a query for TCustomMyStoredProc based on the Params and StoredProcName properties, and assigns it to the SQL property.  |
| <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>RefreshQuick</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )   | Retrieves changes posted to the server by another clients on the client side quickly.  |
| <a href="#"><u>RefreshRecord</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></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="#">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="#">UnLockTable</a> (inherited from <a href="#">TCustomMyDataSet</a> )    | Releases a table locked by the <a href="#">TCustomMyDataSet.LockTable</a> method.  |
| <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.   |

[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

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

© 1997-2012 Devart. All Rights Reserved.

Executes a SQL statement on the server.

### Class

[TCustomMyStoredProc](#)

### Syntax

```
procedure ExecProc;
```

### Remarks

Call the ExecProc method to execute a SQL statement on the server. If SQL statement is a query, ExecProc calls the Open method.

Internally ExecProc calls inherited [TCustomDADataset.Execute](#) method and is only included for compatibility with BDE.

### See Also

- [TCustomDADataset.Execute](#)

© 1997-2012 Devart. All Rights Reserved.

Builds a query for TCustomMyStoredProc based on the Params and StoredProcName properties, and assigns it to the SQL property.

### Class

[TCustomMyStoredProc](#)

### Syntax

```
procedure PrepareSQL;
```

### Remarks

Call the PrepareSQL method to build a query for TCustomMyStoredProc based on the Params and StoredProcName properties, and assign it to the SQL property. Then the generated query is verified to be valid and, if necessary, the list of parameters is modified.

PrepareSQL is called implicitly when TCustomMyStoredProc is executed.

### See Also

- [TCustomDADataset.Params](#)
- [StoredProcName](#)
- [ExecProc](#)

© 1997-2012 Devart. All Rights Reserved.

## 17.15.1.5 MyAccess.TCustomMyTable 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 [TCustomMyTable](#) members.

**Unit**

[MyAccess](#)

**Syntax**

```
TCustomMyTable = class (TCustomMyDataSet) ;
```

**Remarks**

TCustomMyTable implements functionality to access data in a table. Use TCustomMyTable properties and methods to gain direct access to records and fields in an underlying server database without writing SQL statements.

**Inheritance Hierarchy**

```

TObject
  TMemDataSet
    TCustomDADataset
      TCustomMyDataSet
        TCustomMyTable

```

**See Also**

- [TMyTable](#)
- [TCustomMyDataSet](#)
- [TMyQuery](#)
- [Master/Detail Relationships](#)

© 1997-2012 Devart. All Rights Reserved.

[TCustomMyTable](#) class overview.

**Properties**

| Name  | Description  |
|---|--|
| <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="#">CommandTimeout</a> (inherited from <a href="#">TCustomMyDataSet</a> ) | Used to specify the amount of time to attempt execution of a command.  |
| <a href="#">Connection</a> (inherited from <a href="#">TCustomMyDataSet</a> )     | Used to specify a connection object that will be used to connect to a data store.  |
| <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="#">Encryption</a> (inherited from <a href="#">TCustomDADataset</a> )     | Used to specify the options of the data encryption in a dataset.   |
| <a href="#">FetchAll</a> (inherited from <a href="#">TCustomMyDataSet</a> )       | Description is not available at the moment.  |

[FetchRows](#) (inherited from [TCustomDADataset](#))

[FilterSQL](#) (inherited from [TCustomDADataset](#))

[FinalSQL](#) (inherited from [TCustomDADataset](#))

[IndexDefs](#)

[IndexFieldNames](#) (inherited from [TMemDataSet](#))

[InsertId](#) (inherited from [TCustomMyDataSet](#))

[IsQuery](#) (inherited from [TCustomDADataset](#))

[KeyFields](#) (inherited from [TCustomDADataset](#))

[Limit](#)

[LocalConstraints](#) (inherited from [TMemDataSet](#))

[LocalUpdate](#) (inherited from [TMemDataSet](#))

[LockMode](#) (inherited from [TCustomMyDataSet](#))

[MacroCount](#) (inherited from [TCustomDADataset](#))

[Macros](#) (inherited from [TCustomDADataset](#))

[MasterFields](#) (inherited from [TCustomDADataset](#))

[MasterSource](#) (inherited from [TCustomDADataset](#))

[Offset](#)

[Options](#)

[ParamCheck](#) (inherited from [TCustomDADataset](#))

[ParamCount](#) (inherited from [TCustomDADataset](#))

Used to define the number of rows to be transferred across the network at the same time.

Used to change the WHERE clause of SELECT statement and reopen a query.

Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.

Contains information about the indexes for a table.

Used to get or set the list of fields on which the recordset is sorted.

Returns the ID generated for an AUTO INCREMENT column by the previous query.

Used to check whether SQL statement returns rows.

Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.

Used to set the number of rows retrieved from the query.

Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

Used to prevent implicit update of rows on database server.

Specifies when to perform locking of an editing record.

Used to get the number of macros associated with the Macros property.

Makes it possible to change SQL queries easily.

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.

Used to specify the data source component which binds current dataset to the master one.

Used to allow retrieving data from the server starting from the specified row.

Specifies the behaviour of the [TMyTable](#) object.

Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.

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="#">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="#">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="#">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="#">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> (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="#">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="#">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="#">EmptyTable</a>   | Deletes all records from the database table specified by the TableName 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> (inherited from <a href="#">TCustomDADataset</a> )           | Used to learn whether 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="#">TCustomDADataset</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> (inherited from <a href="#">TCustomDADataset</a> )       | Returns internal field types defined in the MemData and accompanying modules.  |
| <a href="#">GetFieldEnum</a> (inherited from <a href="#">TCustomMyDataSet</a> )      | Retrieve the list of acceptable values for a specified field given by the FieldName parameter.   |
| <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="#">TCustomMyDataSet</a> )              | Overloaded. Locks the current record for the current connection.   |



|  |   |
|--|---|
| <a href="#">LockTable</a> (inherited from <a href="#">TCustomMyDataSet</a> )     | Locks table for the current connection.   |
| <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="#">TCustomDADataset</a> )   | Sets or uses 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="#">RefreshQuick</a> (inherited from <a href="#">TCustomMyDataSet</a> )  | Retrieves changes posted to the server by another clients on the client side quickly.                                   |
| <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="#">UnLockTable</a> (inherited from <a href="#">TCustomMyDataSet</a> )   | Releases a table locked by the <a href="#">TCustomMyDataSet.LockTable</a> method.                                       |
| <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.                     |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TCustomMyTable** class.

For a complete list of the **TCustomMyTable** class members, see the [TCustomMyTable 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="#">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="#">CommandTimeout</a> (inherited from <a href="#">TCustomMyDataSet</a> )      | Used to specify the amount of time to attempt execution of a command.  |
| <a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )            | Clears the cached updates buffer.  |
| <a href="#">Connection</a> (inherited from <a href="#">TCustomMyDataSet</a> )          | Used to specify a connection object that will be used 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="#"><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>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>FetchAll</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )     | Description is not available at the moment.  |
| <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="#"><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>TCustomDADataset</u></a> )    | Determines if a parameter with the specified name exists in a dataset.   |
| <a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )           | Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself 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.  |

[GetFieldEnum](#) (inherited from [TCustomMyDataSet](#))

[GetFieldObject](#) (inherited from [TCustomDADataset](#))

[GetFieldPrecision](#) (inherited from [TCustomDADataset](#))

[GetFieldScale](#) (inherited from [TCustomDADataset](#))

[GetOrderBy](#) (inherited from [TCustomDADataset](#))

[GotoCurrent](#) (inherited from [TCustomDADataset](#))

[IndexDefs](#)

[IndexFieldNames](#) (inherited from [TMemDataSet](#))

[InsertId](#) (inherited from [TCustomMyDataSet](#))

[IsQuery](#) (inherited from [TCustomDADataset](#))

[KeyFields](#) (inherited from [TCustomDADataset](#))

[Limit](#)

[LocalConstraints](#) (inherited from [TMemDataSet](#))

[LocalUpdate](#) (inherited from [TMemDataSet](#))

[Locate](#) (inherited from [TMemDataSet](#))

[LocateEx](#) (inherited from [TMemDataSet](#))

[Lock](#) (inherited from [TCustomMyDataSet](#))

[LockMode](#) (inherited from [TCustomMyDataSet](#))

[LockTable](#) (inherited from [TCustomMyDataSet](#))

[MacroByName](#) (inherited from [TCustomDADataset](#))

[MacroCount](#) (inherited from [TCustomDADataset](#))

[Macros](#) (inherited from [TCustomDADataset](#))

Retrieve the list of acceptable values for a specified field given by the FieldName parameter.

Returns a multireference shared object from field.

Retrieves the precision of a number field.

Retrieves the scale of a number field.

Retrieves an ORDER BY clause from a SQL statement.

Sets the current record in this dataset similar to the current record in another dataset.

Contains information about the indexes for a table.

Used to get or set the list of fields on which the recordset is sorted.

Returns the ID generated for an AUTO INCREMENT column by the previous query.

Used to check whether SQL statement returns rows.

Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.

Used to set the number of rows retrieved from the query.

Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

Used to prevent implicit update of rows on database server.

Overloaded. Searches a dataset for a specific record and positions the cursor on it.

Overloaded. Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet.

Overloaded. Locks the current record for the current connection.

Specifies when to perform locking of an editing record.

Locks table for the current connection.

Finds a Macro with the name passed in Name.

Used to get the number of macros associated with the Macros property.

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="#">Offset</a>  | Used to allow retrieving data from the server starting from the specified row.   |
| <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>   | Specifies the behaviour of the <a href="#">TMyTable</a> 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="#">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="#">RefreshQuick</a> (inherited from <a href="#">TCustomMyDataSet</a> )   | Retrieves changes posted to the server by another clients on the client side quickly.  |
| <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.  |

[RowsAffected](#) (inherited from [TCustomDADataset](#))

[SaveSQL](#) (inherited from [TCustomDADataset](#))

[SaveToXML](#) (inherited from [TMemDataSet](#))

[SetOrderBy](#) (inherited from [TCustomDADataset](#))

[SQL](#) (inherited from [TCustomDADataset](#))

[SQLDelete](#) (inherited from [TCustomDADataset](#))

[SQLInsert](#) (inherited from [TCustomDADataset](#))

[SQLLock](#) (inherited from [TCustomDADataset](#))

[SQLRefresh](#) (inherited from [TCustomDADataset](#))

[SQLSaved](#) (inherited from [TCustomDADataset](#))

[SQLUpdate](#) (inherited from [TCustomDADataset](#))

[UniDirectional](#) (inherited from [TCustomDADataset](#))

[UnLock](#) (inherited from [TCustomDADataset](#))

[UnLockTable](#) (inherited from [TCustomMyDataSet](#))

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

[UpdatesPending](#) (inherited from [TMemDataSet](#))

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

Saves the SQL property value to BaseSQL.

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Builds an ORDER BY clause of a SELECT statement.

Used to provide a SQL statement that a query component executes when its Open method is called.

Used to specify a SQL statement that will be used when applying a deletion to a record.

Used to specify the SQL statement that will be used when applying an insertion to a dataset.

Used to specify a SQL statement that will be used to perform a record lock.

Used to specify a SQL statement that will be used to refresh current record by calling the [TCustomDADataset.RefreshRecord](#) procedure.

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

Used to specify a SQL statement that will be used when applying an update to a dataset.

Used if an application does not need bidirectional access to records in the result set.

Releases a record lock.

Releases a table locked by the [TCustomMyDataSet.LockTable](#) method.

Frees the resources allocated for a previously prepared query on the server and client sides.

Used to indicate the update status for the current record when cached updates are enabled.

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

Used to check the status of the cached updates buffer.

Indicates the current update status for the dataset when cached updates are enabled.

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Contains information about the indexes for a table.

### Class

[TCustomMyTable](#)

### Syntax

```
property IndexDefs: TIndexDefs;
```

### Remarks

The IndexDefs property is used to contain information about the indexes for a table. IndexDefs is a collection of index definitions, each of which describes an available index for the table. In contrast to BDE, can be used only for viewing the list of indexes already created for the table. As an additional request for the server is required to fill it, IndexDefs is filled on the first call.

© 1997-2012 Devart. All Rights Reserved.

Used to set the number of rows retrieved from the query.

### Class

[TCustomMyTable](#)

### Syntax

```
property Limit: integer default - 1;
```

### Remarks

Use the Limit property to set the number of rows retrieved from the query. If Limit is -1, all records will be obtained.

### See Also

- [Offset](#)

© 1997-2012 Devart. All Rights Reserved.

Used to allow retrieving data from the server starting from the specified row.

### Class

[TCustomMyTable](#)

### Syntax

```
property Offset: integer default 0;
```

### Remarks

Use the Offset property to allow retrieving data from the server starting from the specified row. The default value is 0.

### See Also

- [Limit](#)

© 1997-2012 Devart. All Rights Reserved.

Specifies the behaviour of the [TMyTable](#) object.

### Class

[TCustomMyTable](#)

## Syntax

**property** Options: [TMyTableOptions](#);

## Remarks

Set the properties of Options to specify the behaviour of a [TMyTable](#) object. Descriptions of all options are in the table below.

| Option Name                  | Description  |
|------------------------------|--|
| <a href="#">HandlerIndex</a> | Used to assign an index and a value that this index should satisfy.        |
| <a href="#">UseHandler</a>   | Used for the HANDLER statement to be used instead of the SELECT statement. |

©

1997-2012 Devart. All Rights Reserved.

Methods of the **TCustomMyTable** class.

For a complete list of the **TCustomMyTable** class members, see the [TCustomMyTable 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="#">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="#">CommandTimeout</a> (inherited from <a href="#">TCustomMyDataSet</a> )      | Used to specify the amount of time to attempt execution of a command.                         |
| <a href="#">CommitUpdates</a> (inherited from <a href="#">TMemDataSet</a> )            | Clears the cached updates buffer.   |
| <a href="#">Connection</a> (inherited from <a href="#">TCustomMyDataSet</a> )          | Used to specify a connection object that will be used 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="#">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="#">EmptyTable</a>  | Deletes all records from the database table specified by the TableName property.   |
| <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="#">FetchAll</a> (inherited from <a href="#">TCustomMyDataSet</a> )         | Description is not available at the moment.  |
| <a href="#">Fetched</a> (inherited from <a href="#">TCustomDADataset</a> )          | Used to learn whether 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="#">TCustomDADataset</a> )        | Determines if a parameter with the specified name exists in a dataset.   |

|  |  |
|--|--|
| <a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )                | Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself 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>GetFieldEnum</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Retrieve the list of acceptable values for a specified field given by the FieldName parameter.                                       |
| <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>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>InsertId</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )          | Returns the ID generated for an AUTO INCREMENT column by the previous query.   |
| <a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )           | Used to check whether 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>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>TCustomMyDataSet</u></a> )              | Overloaded. Locks the current record for the current connection.   |
| <a href="#"><u>LockMode</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )          | Specifies when to perform locking of an editing record.  |
| <a href="#"><u>LockTable</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )         | Locks table for the current connection.  |
| <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>TCustomMyDataSet</u></a> )        | Specifies the behaviour of TCustomMyDataSet object.  |
| <a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )    | Sets or uses 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>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>RefreshQuick</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )   | Retrieves changes posted to the server by another clients on the client side quickly.  |
| <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="#"><u>Resync</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )         | Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.  |
| <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>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>SaveSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )        | Saves the SQL property value to BaseSQL.   |

[SaveToXML](#) (inherited from [TMemDataSet](#))

[SetOrderBy](#) (inherited from [TCustomDADataset](#))

[SQL](#) (inherited from [TCustomDADataset](#))

[SQLDelete](#) (inherited from [TCustomDADataset](#))

[SQLInsert](#) (inherited from [TCustomDADataset](#))

[SQLLock](#) (inherited from [TCustomDADataset](#))

[SQLRefresh](#) (inherited from [TCustomDADataset](#))

[SQLSaved](#) (inherited from [TCustomDADataset](#))

[SQLUpdate](#) (inherited from [TCustomDADataset](#))

[UniDirectional](#) (inherited from [TCustomDADataset](#))

[UnLock](#) (inherited from [TCustomDADataset](#))

[UnLockTable](#) (inherited from [TCustomMyDataSet](#))

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

[UpdatesPending](#) (inherited from [TMemDataSet](#))

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Builds an ORDER BY clause of a SELECT statement.

Used to provide a SQL statement that a query component executes when its Open method is called.

Used to specify a SQL statement that will be used when applying a deletion to a record.

Used to specify the SQL statement that will be used when applying an insertion to a dataset.

Used to specify a SQL statement that will be used to perform a record lock.

Used to specify a SQL statement that will be used to refresh current record by calling the [TCustomDADataset.RefreshRecord](#) procedure.

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

Used to specify a SQL statement that will be used when applying an update to a dataset.

Used if an application does not need bidirectional access to records in the result set.

Releases a record lock.

Releases a table locked by the [TCustomMyDataSet.LockTable](#) method.

Frees the resources allocated for a previously prepared query on the server and client sides.

Used to indicate the update status for the current record when cached updates are enabled.

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

Used to check the status of the cached updates buffer.

Indicates the current update status for the dataset when cached updates are enabled.

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Deletes all records from the database table specified by the TableName property.

## Class

[TCustomMyTable](#)

## Syntax

```
procedure EmptyTable;
```

## Remarks

Call the EmptyTable method to delete all records from the database table specified by the TableName property.

## See Also

- P:Devart.MyDac.TCustomMyTable.TableName

© 1997-2012 Devart. All Rights Reserved.

### 17.15.1.6 MyAccess.TMyCommand Class

A component for execution of SQL statements and stored procedures which do not return rowsets. For a list of all members of this type, see [TMyCommand](#) members.

## Unit

[MyAccess](#)

## Syntax

```
TMyCommand = class (TCustomDASQL) ;
```

## Remarks

Use TMyCommand to access database server using SQL statements.

TMyCommand object in a client application is used mainly to execute SQL statements on database server. SQL statement should not retrieve records from a database since TMyCommand does not provide storage for returned data.

## Inheritance Hierarchy

TObject

[TCustomDASQL](#)

**TMyCommand**

## See Also

- [TMyQuery](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyCommand](#) 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="#">CommandTimeout</a>  | Used to specify the amount of time to attempt to execute a command.                          |
| <a href="#">Connection</a>  | Used to specify a connection object that will be used to connect to a data store.            |
| <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="#">FinalSQL</a> (inherited from <a href="#">TCustomDASQL</a> )     | Used to return a SQL statement with expanded macros.   |

[InsertId](#)

Returns the ID generated for an AUTO INCREMENT column by the previous query.

[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](#) (inherited from [TCustomDASQL](#))

Used to contain parameters for a 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.

## Methods

| Name   | Description   |
|--|---|
| <a href="#">BreakExec</a>  | Breaks execution of the SQL statement on the server.  |
| <a href="#">Execute</a> (inherited from <a href="#">TCustomDASQL</a> )       | Overloaded. Executes SQL commands.  |
| <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> (inherited from <a href="#">TCustomDASQL</a> )     | Finds 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> (inherited from <a href="#">TCustomDASQL</a> )   | Finds 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. |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyCommand** class.

For a complete list of the **TMyCommand** class members, see the [TMyCommand 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="#">ChangeCursor</a> (inherited from <a href="#">TCustomDASQL</a> ) | Enables or disables changing screen cursor when executing commands in the NonBlocking mode.                                 |
| <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="#">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> (inherited from <a href="#">TCustomDASQL</a> )    | Finds a parameter with the specified name.  |
| <a href="#">InsertId</a>  | Returns the ID generated for an AUTO INCREMENT column by the previous query.  |
| <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> (inherited from <a href="#">TCustomDASQL</a> )  | Finds 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.                   |

[UnPrepare](#) (inherited from [TCustomDASQL](#))

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.

## Published

### Name

[CommandTimeout](#)

### Description

Used to specify the amount of time to attempt to execute a command.

[Connection](#)

Used to specify a connection object that will be used to connect to a data store.

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Used to specify the amount of time to attempt to execute a command.

## Class

[TMyCommand](#)

## Syntax

```
property CommandTimeout: integer default 0;
```

## Remarks

Use the CommandTimeout property to specify the amount of time to attempt to execute a command.

Use CommandTimeout to specify the amount of time that expires before an attempt to execute a command is considered unsuccessful. Measured in seconds.

If a command executed successfully prior to the expiration of the seconds specified, CommandTimeout has no effect.

In the case of exceeding waiting time error CR\_SERVER\_LOST 'Lost connection to MySQL server during query' raises.

The default value is 0 (infinite).

## See Also

- [TCustomMyConnection.ConnectionTimeout](#)
- [TCustomMyDataSet.CommandTimeout](#)

© 1997-2012 Devart. All Rights Reserved.

Used to specify a connection object that will be used to connect to a data store.

## Class

[TMyCommand](#)

## Syntax

```
property Connection: TCustomMyConnection;
```

## 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 TMyConnection objects.

At run-time, set the Connection property to reference an existing TMyConnection object.

## See Also



- [TCustomMyConnection](#)

© 1997-2012 Devart. All Rights Reserved.

Returns the ID generated for an AUTO INCREMENT column by the previous query.

## Class

[TMyCommand](#)

## Syntax

**property** InsertId: int64;

## Remarks

Use the InsertId to return the ID generated for an AUTO INCREMENT column by the previous query. Use this function after you have performed an INSERT query into a table that contains an AUTO INCREMENT field. If the query does not perform an insertion into a table that contains an AUTO INCREMENT field the value of InsertId won't be defined.

## See Also

- [TCustomDASQL.Execute](#)

© 1997-2012 Devart. All Rights Reserved.

Methods of the **TMyCommand** class.

For a complete list of the **TMyCommand** class members, see the [TMyCommand 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 the 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="#">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="#">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> (inherited from <a href="#">TCustomDASQL</a> )    | Finds 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.                        |

[Macros](#) (inherited from [TCustomDASQL](#))

[ParamByName](#) (inherited from [TCustomDASQL](#))

[ParamCheck](#) (inherited from [TCustomDASQL](#))

[ParamCount](#) (inherited from [TCustomDASQL](#))

[Params](#) (inherited from [TCustomDASQL](#))

[ParamValues](#) (inherited from [TCustomDASQL](#))

[Prepare](#) (inherited from [TCustomDASQL](#))

[Prepared](#) (inherited from [TCustomDASQL](#))

[RowsAffected](#) (inherited from [TCustomDASQL](#))

[SQL](#) (inherited from [TCustomDASQL](#))

[UnPrepare](#) (inherited from [TCustomDASQL](#))

[WaitExecuting](#) (inherited from [TCustomDASQL](#))

Makes it possible to change SQL queries easily.

Finds a parameter with the specified name.

Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.

Indicates the number of parameters in the Params property.

Used to contain parameters for a SQL statement.

Used to get or set the values of individual field parameters that are identified by name.

Allocates, opens, and parses cursor for a query.

Used to indicate whether a query is prepared for execution.

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.

Frees the resources allocated for a previously prepared query on the server and client sides.

Waits until TCustomDASQL executes a SQL statement.

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Breaks execution of the SQL statement on the server.

## Class

[TMyCommand](#)

## Syntax

```
procedure BreakExec;
```

## Remarks

Call the BreakExec method to break execution of the SQL statement on the server. Execution is broken by the KILL operator execution on server. It makes sense to call BreakExec only from another thread.

## See Also

- [TCustomDASQL.Execute](#)
- [TCustomMyDataSet.BreakExec](#)

© 1997-2012 Devart. All Rights Reserved.

## 17.15.1.7 MyAccess.TMyConnection Class

A component for setting up and controlling connections to MySQL database server.

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

**Unit**

[MyAccess](#)

**Syntax**

```
TMyConnection = class(TCustomMyConnection);
```

**Remarks**

TMyConnection component is used to establish connection to database server, provide customized login support, and perform transaction control. TMyConnection publishes connection-related properties derived from its ancestor class TCustomMyConnection and introduces specific properties.

**Inheritance Hierarchy**

TObject

[TCustomDAConnection](#)

[TCustomMyConnection](#)

**TMyConnection**

**See Also**

- [TCustomMyConnection](#)
- [TMyEmbConnection](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyConnection](#) class overview.

**Properties**

| Name  | Description  |
|---|--|
| <a href="#">ClientVersion</a> (inherited from <a href="#">TCustomMyConnection</a> )     | Contains the version of the MySQL Client library.  |
| <a href="#">ConnectDialog</a> (inherited from <a href="#">TCustomDAConnection</a> )     | Allows to link a <a href="#">TCustomConnectDialog</a> component.   |
| <a href="#">ConnectionTimeout</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Used to specify the amount of time to attempt to establish a connection.   |
| <a href="#">ConvertEOL</a> (inherited from <a href="#">TCustomDAConnection</a> )        | Allows customizing line breaks in string fields and parameters.  |
| <a href="#">Database</a> (inherited from <a href="#">TCustomMyConnection</a> )          | Used to specify a database name that is a default source of data for SQL queries once a connection is established.     |
| <a href="#">HttpOptions</a>   | Holds a THttpOptions object that contains settings for HTTP connection.  |
| <a href="#">InTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )     | Indicates whether the transaction is active.   |
| <a href="#">IOHandler</a>   | Used to assign an external component for communication between MyDAC and MySQL server.                                 |
| <a href="#">IsolationLevel</a> (inherited from <a href="#">TCustomMyConnection</a> )    | Used to specify extent to which all outside transactions interfere with subsequent transactions of current connection. |
| <a href="#">LoginPrompt</a> (inherited from <a href="#">TCustomDAConnection</a> )       | Specifies whether a login dialog appears immediately before opening a new connection.                                  |

[Options](#)[Password](#) (inherited from [TCustomDAConnection](#))[Pooling](#) (inherited from [TCustomDAConnection](#))[PoolingOptions](#) (inherited from [TCustomDAConnection](#))[Port](#)[Server](#) (inherited from [TCustomDAConnection](#))[ServerVersion](#) (inherited from [TCustomMyConnection](#))[SSLOptions](#)[ThreadId](#) (inherited from [TCustomMyConnection](#))[Username](#) (inherited from [TCustomDAConnection](#))

Specifies the behaviour of TMyConnection object.

Serves to supply a password for login.

Enables or disables using connection pool.

Specifies the behaviour of connection pool.

Used to specify the port number for TCP/IP connection.

Serves to supply the server name for login.

Holds the version of MySQL server.

Used to set the properties required for protected SSL connection with the server.

Used to return the thread ID of the current connection.

Used to supply a user name for login.

**Methods****Name****Description**[ApplyUpdates](#) (inherited from [TCustomDAConnection](#))

Overloaded. Applies changes in datasets.

[AssignConnect](#) (inherited from [TCustomMyConnection](#))

Shares database connection between the TCustomMyConnection components.

[Commit](#) (inherited from [TCustomDAConnection](#))

Commits current transaction.

[Connect](#) (inherited from [TCustomDAConnection](#))

Establishes a connection to the server.

[CreateDataSet](#) (inherited from [TCustomMyConnection](#))

Returns a new instance of TCustomMyDataSet class and associates it with this connection object.

[CreateSQL](#) (inherited from [TCustomDAConnection](#))

Creates a component for queries execution.

[Disconnect](#) (inherited from [TCustomDAConnection](#))

Performs disconnect.

[ExecProc](#) (inherited from [TCustomDAConnection](#))

Allows to execute stored procedure or function providing its name and parameters.

[ExecProcEx](#) (inherited from [TCustomDAConnection](#))

Allows to execute a stored procedure or function.

[ExecSQL](#) (inherited from [TCustomMyConnection](#))Executes any SQL statement outside [TMyQuery](#) or [TMyCommand](#) components.[ExecSQLEx](#) (inherited from [TCustomDAConnection](#))

Executes any SQL statement outside the TQuery or TSQL components.

[GetCharsetNames](#) (inherited from [TCustomMyConnection](#))

Populates a string list with the names of available charsets.

[GetDatabaseNames](#) (inherited from [TCustomDAConnection](#))

Returns a database list from the server.

[GetExecuteInfo](#) (inherited from [TCustomMyConnection](#))

Returns the result of the last query execution.

[GetStoredProcNames](#) (inherited from [TCustomDAConnection](#))

Returns a list of stored procedures from the server.

[GetTriggerNames](#) (inherited from [TCustomMyConnection](#))

Returns a list of triggers from the server.

|   |   |
|---|---|
| <a href="#">MonitorMessage</a> (inherited from <a href="#">TCustomDAConnection</a> )      | Sends a specified message through the <a href="#">TCustomDASQLMonitor</a> component.                    |
| <a href="#">Ping</a> (inherited from <a href="#">TCustomMyConnection</a> )                | Allows to avoid automatic disconnection of the client by the server.                                    |
| <a href="#">ReleaseSavepoint</a> (inherited from <a href="#">TCustomMyConnection</a> )    | Releases the specified savepoint without affecting any work that has been performed after its creation. |
| <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="#">RollbackToSavepoint</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Cancels all updates for the current transaction.  |
| <a href="#">Savepoint</a> (inherited from <a href="#">TCustomMyConnection</a> )           | Defines a point in the transaction to which you can roll back later.                                    |
| <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.                   |
| <a href="#">OnError</a> (inherited from <a href="#">TCustomDAConnection</a> )          | This event occurs when an error has arisen in the connection. |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyConnection** class.

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

## Public

| Name  | Description   |
|---|---|
| <a href="#">ApplyUpdates</a> (inherited from <a href="#">TCustomDAConnection</a> )      | Overloaded. Applies changes in datasets.  |
| <a href="#">AssignConnect</a> (inherited from <a href="#">TCustomMyConnection</a> )     | Shares database connection between the TCustomMyConnection components.                          |
| <a href="#">ClientVersion</a> (inherited from <a href="#">TCustomMyConnection</a> )     | Contains the version of the MySQL Client library.   |
| <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="#">ConnectionTimeout</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Used to specify the amount of time to attempt to establish a connection.                        |
| <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="#">TCustomMyConnection</a> )     | Returns a new instance of TCustomMyDataSet class and associates it with this connection object. |
| <a href="#">CreateSQL</a> (inherited from <a href="#">TCustomDAConnection</a> )         | Creates a component for queries execution.  |

|  |  |
|--|--|
| <a href="#"><u>Database</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )           | Used to specify a database name that is a default source of data for SQL queries once a connection is established.     |
| <a href="#"><u>Disconnect</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )         | Performs disconnect.   |
| <a href="#"><u>ExecProc</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )           | Allows to execute stored procedure or function providing its name and parameters.                                      |
| <a href="#"><u>ExecProcEx</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )         | Allows to execute a stored procedure or function.  |
| <a href="#"><u>ExecSQL</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )            | Executes any SQL statement outside <a href="#"><u>TMyQuery</u></a> or <a href="#"><u>TMyCommand</u></a> components.    |
| <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>GetCharsetNames</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )    | Populates a string list with the names of available charsets.  |
| <a href="#"><u>GetDatabaseNames</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )   | Returns a database list from the server.   |
| <a href="#"><u>GetExecuteInfo</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )     | Returns the result of the last query execution.  |
| <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>GetTriggerNames</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )    | Returns a list of triggers from the server.  |
| <a href="#"><u>InTransaction</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )      | Indicates whether the transaction is active.   |
| <a href="#"><u>IsolationLevel</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )     | Used to specify extent to which all outside transactions interfere with subsequent transactions of current connection. |
| <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>Password</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )           | Serves to supply a password for login.   |
| <a href="#"><u>Ping</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )               | Allows to avoid automatic disconnection of the client by the server.   |
| <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>ReleaseSavepoint</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )   | Releases the specified savepoint without affecting any work that has been performed after its creation.                |
| <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>Rollback</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )           | Discards all current data changes and ends transaction.  |

|   |  |
|---|--|
| <a href="#">RollbackToSavepoint</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Cancels all updates for the current transaction.                     |
| <a href="#">Savepoint</a> (inherited from <a href="#">TCustomMyConnection</a> )           | Defines a point in the transaction to which you can roll back later. |
| <a href="#">Server</a> (inherited from <a href="#">TCustomDAConnection</a> )              | Serves to supply the server name for login.                          |
| <a href="#">ServerVersion</a> (inherited from <a href="#">TCustomMyConnection</a> )       | Holds the version of MySQL server.                                   |
| <a href="#">StartTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )    | Begins a new user transaction.                                       |
| <a href="#">ThreadId</a> (inherited from <a href="#">TCustomMyConnection</a> )            | Used to return the thread ID of the current connection.              |
| <a href="#">Username</a> (inherited from <a href="#">TCustomDAConnection</a> )            | Used to supply a user name for login.                                |

## Published

| Name                        | Description  |
|-----------------------------|--|
| <a href="#">HttpOptions</a> | Holds a THttpOptions object that contains settings for HTTP connection.                |
| <a href="#">IOHandler</a>   | Used to assign an external component for communication between MyDAC and MySQL server. |
| <a href="#">Options</a>     | Specifies the behaviour of TMyConnection object.                                       |
| <a href="#">Port</a>        | Used to specify the port number for TCP/IP connection.                                 |
| <a href="#">SSLOptions</a>  | Used to set the properties required for protected SSL connection with the server.      |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Holds a THttpOptions object that contains settings for HTTP connection.

## Class

[TMyConnection](#)

## Syntax

```
property HttpOptions: THttpOptions;
```

## Remarks

The HttpOptions property holds a THttpOptions object that contains settings for HTTP connection. For more information on HTTP tunneling refer to the [Network Tunneling](#) article.

© 1997-2012 Devart. All Rights Reserved.

Used to assign an external component for communication between MyDAC and MySQL server.

## Class

[TMyConnection](#)

## Syntax

```
property IOHandler: TCRIOHandler;
```

## Remarks

Use the IOHandler property to assign an external component for communication between MyDAC and

MySQL server. The component must be a descendant of the TMyIOHandler abstract class. There is an example of implementation and usage of such component in the SecureBridge demo. This component provides integration with the [SecureBridge](#) library, so [SecureBridge](#) should be also installed to build and install it. You can read more about this demo in the [Demo Projects](#) topic, and in the [Readme.html](#) file located in the the SecureBridge demo directory. [SecureBridge](#) is a library that can be used for ensuring protection of important data transferred between MySQL server and MyDAC based applications through public networks.

## See Also

- [TMyIOHandler](#)

© 1997-2012 Devart. All Rights Reserved.

Specifies the behaviour of TMyConnection object.

## Class

[TMyConnection](#)

## Syntax

**property** Options: [TMyConnectionOptions](#);

## Remarks

Set the properties of Options to specify the behaviour of a TMyConnection object. Descriptions of all options are in the table below.

| Option Name                      | Description  |
|----------------------------------|--|
| <a href="#">CheckBackslashes</a> | Used to check the value of the NO BACKSLASH ESCAPES server variable. |
| <a href="#">Compress</a>         | Used to apply compression on transferring data.                      |
| <a href="#">Direct</a>           | Used to work without using MySQL client library (libmysql.dll).      |
| <a href="#">Embedded</a>         | Used to specify the server that will be used to connect.             |
| <a href="#">Interactive</a>      | Used to make a connection interactive.                               |
| <a href="#">Protocol</a>         | Used to specify the protocol to use when connecting to server.       |

## See Also

- [TCustomDAConnection.Server](#)
- [TCustomMyConnection.Database](#)
- [TCustomMyConnection.Options](#)
- [Embedded Server](#)

©

1997-2012 Devart. All Rights Reserved.



Used to specify the port number for TCP/IP connection.

## Class

[TMyConnection](#)

## Syntax

```
property Port: integer default MYSQL_PORT;
```

## Remarks

Use the Port property to specify the port number for TCP/IP connection. Note that [TCustomDAConnection.Server](#) property determines the type of the connection.

The default value is 3306.

The Port property can be used only if [TCustomMyConnection.Options](#) is set to False.

## See Also

- [TCustomDAConnection.Server](#)
- [TCustomMyConnection.Database](#)

© 1997-2012 Devart. All Rights Reserved.

Used to set the properties required for protected SSL connection with the server.

## Class

[TMyConnection](#)

## Syntax

```
property SSLOptions: TMyConnectionSSLOptions;
```

## Remarks

Use the SSLOptions property to set the properties required for protected SSL connection with the server. These properties can be used only for [Options](#) = mpSSL.

For using ssleay32.dll and libeay32.dll files are needed.

The detailed description of these properties you can find in MySQL Reference Manual

Descriptions of all options are in the table below.

| Option Name                 | Description  |
|-----------------------------|--|
| <a href="#">CACert</a>      | Holds the pathname to the certificate authority file.        |
| <a href="#">Cert</a>        | Holds the pathname to the certificate file.                  |
| <a href="#">ChipherList</a> | Holds the list of allowed ciphers to use for SSL encryption. |
| <a href="#">Key</a>         | Holds the pathname to the key file.                          |

## See Also

- [Options](#)

©

1997-2012 Devart. All Rights Reserved.

### 17.15.1.8 MyAccess.TMyConnectionOptions Class

This class allows setting up the behaviour of the TMyConnection class.

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

## Unit

[MyAccess](#)

## Syntax

```
TMyConnectionOptions = class (TCustomMyConnectionOptions) ;
```

## Inheritance Hierarchy

```

TObject
  TDAConnectionOptions
    TCustomMyConnectionOptions
      TMyConnectionOptions

```

© 1997-2012 Devart. All Rights Reserved.

[TMyConnectionOptions](#) class overview.

## Properties

| Name   | Description   |
|--|---|
| <a href="#">Charset</a> (inherited from <a href="#">TCustomMyConnectionOptions</a> )               | Used to set a character set used by the client.   |
| <a href="#">CheckBackslashes</a>   | Used to check the value of the NO · BACKSLASH · ESCAPES server variable.  |
| <a href="#">Compress</a>   | Used to apply compression on transferring data.   |
| <a href="#">DefaultSortType</a> (inherited from <a href="#">TDAConnectionOptions</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</a> . <a href="#">IndexFieldNames</a> property of a dataset. |
| <a href="#">Direct</a>   | Used to work without using MySQL client library (libmysql.dll).   |
| <a href="#">DisconnectedMode</a> (inherited from <a href="#">TDAConnectionOptions</a> )            | Used to open a connection only when needed for performing a server call and closes after performing the operation.  |
| <a href="#">Embedded</a>   | Used to specify the server that will be used to connect.  |
| <a href="#">Interactive</a>  | Used to make a connection interactive.  |
| <a href="#">KeepDesignConnected</a> (inherited from <a href="#">TDAConnectionOptions</a> )         | Used to prevent an application from establishing a connection at the time of startup.   |
| <a href="#">LocalFailover</a> (inherited from <a href="#">TDAConnectionOptions</a> )               | If True, the <a href="#">TCustomDAConnection.OnConnectionLost</a> event occurs and a failover operation can be performed after connection breaks.   |
| <a href="#">NullForZeroDelphiDate</a> (inherited from <a href="#">TCustomMyConnectionOptions</a> ) | Used to hide the '30-12-1899' dates.  |
| <a href="#">NumericType</a> (inherited from <a href="#">TCustomMyConnectionOptions</a> )           | Used to specify the format of storing and representation of the NUMERIC (DECIMAL) fields for all <a href="#">TCustomMyDataSets</a> , associated with the given connection.  |
| <a href="#">OptimisedBigInt</a> (inherited from <a href="#">TCustomMyConnectionOptions</a> )       | Used to convert all fields with field length less than 11 of TLargeIntField type into TIntegerField.  |

[Protocol](#)

Used to specify the protocol to use when connecting to server.

[UseUnicode](#) (inherited from [TCustomMyConnectionOptions](#))

Used to inform server that all data between client and server sides will be passed in Utf8 coding.

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyConnectionOptions** class.

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

## Public

| Name   | Description   |
|--|---|
| <a href="#">Charset</a> (inherited from <a href="#">TCustomMyConnectionOptions</a> )               | Used to set a character set used by the client.   |
| <a href="#">DefaultSortType</a> (inherited from <a href="#">TDAConnectionOptions</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="#">TDAConnectionOptions</a> )            | Used to open a connection only when needed for performing a server call and closes after performing the operation.  |
| <a href="#">KeepDesignConnected</a> (inherited from <a href="#">TDAConnectionOptions</a> )         | Used to prevent an application from establishing a connection at the time of startup.   |
| <a href="#">LocalFailover</a> (inherited from <a href="#">TDAConnectionOptions</a> )               | If True, the <a href="#">TCustomDAConnection.OnConnectionLost</a> event occurs and a failover operation can be performed after connection breaks.   |
| <a href="#">NullForZeroDelphiDate</a> (inherited from <a href="#">TCustomMyConnectionOptions</a> ) | Used to hide the '30-12-1899' dates.  |
| <a href="#">NumericType</a> (inherited from <a href="#">TCustomMyConnectionOptions</a> )           | Used to specify the format of storing and representation of the NUMERIC (DECIMAL) fields for all <a href="#">TCustomMyDataSets</a> , associated with the given connection.  |
| <a href="#">OptimizedBigInt</a> (inherited from <a href="#">TCustomMyConnectionOptions</a> )       | Used to convert all fields with field length less than 11 of TLargeIntField type into TIntegerField.  |
| <a href="#">UseUnicode</a> (inherited from <a href="#">TCustomMyConnectionOptions</a> )            | Used to inform server that all data between client and server sides will be passed in Utf8 coding.  |

## Published

| Name                             | Description  |
|----------------------------------|--|
| <a href="#">CheckBackslashes</a> | Used to check the value of the NO BACKSLASH ESCAPES server variable. |
| <a href="#">Compress</a>         | Used to apply compression on transferring data.                      |
| <a href="#">Direct</a>           | Used to work without using MySQL client library (libmysql.dll).      |
| <a href="#">Embedded</a>         | Used to specify the server that will be used to connect.             |

[Interactive](#)

Used to make a connection interactive.

[Protocol](#)

Used to specify the protocol to use when connecting to server.

### See Also

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

© 1997-2012 Devart. All Rights Reserved.

Used to check the value of the NO BACKSLASH ESCAPES server variable.

### Class

[TMyConnectionOptions](#)

### Syntax

```
property CheckBackslashes: boolean default False;
```

### Remarks

Use the CheckBackslashes property to check the value of the NO BACKSLASH ESCAPES server variable. It enables or disables the usage of the backslash character ('\') as an escape character within strings. The backslash character is used when transferring parameters in a query and when dumping a database to a SQL script with the TMyDump component. If the CheckBackslashes property is set to True, the value of the NO BACKSLASH ESCAPES variable is read from the server when establishing a connection. The received value will determine if the backslash character is used as an escape character within strings. If the CheckBackslashes property is set to False, the backslash character will be used in any case. The default value is False.

© 1997-2012 Devart. All Rights Reserved.

Used to apply compression on transferring data.

### Class

[TMyConnectionOptions](#)

### Syntax

```
property Compress: boolean default False;
```

### Remarks

Use the Compress property to use compression on transferring data. Setting this property to True is quite effective on transferring big volumes of data through slow connection. Pay attention that each row is compressed separately. Be careful when setting this option as in some cases it may decrease fetch speed instead of increasing. This property is ignored under CLR. The default value is False.

© 1997-2012 Devart. All Rights Reserved.

Used to work without using MySQL client library (libmysql.dll).

### Class

[TMyConnectionOptions](#)

### Syntax

```
property Direct: boolean default True;
```

### Remarks

Use the Direct property to work without using MySQL client library (libmysql.dll). Used only if Embedded is disabled.

© 1997-2012 Devart. All Rights Reserved.

Used to specify the server that will be used to connect.

## Class

[TMyConnectionOptions](#)

## Syntax

```
property Embedded: boolean default False;
```

## Remarks

Use the Embedded property to specify what server will be used to connect - MySQL server or Embedded MySQL server. You can read about the features and using Embedded server at the [Embedded Server](#). In most cases, you should use [TMyEmbConnection](#) instead of this option.

© 1997-2012 Devart. All Rights Reserved.

Used to make a connection interactive.

## Class

[TMyConnectionOptions](#)

## Syntax

```
property Interactive: boolean default False;
```

## Remarks

If a connection is interactive, MySQL uses the *interactive timeout* MySQL system variable for the number of seconds the server waits for activity on the connection before closing it. Otherwise, MySQL uses the *wait timeout* MySQL system variable for the same purpose.

© 1997-2012 Devart. All Rights Reserved.

Used to specify the protocol to use when connecting to server.

## Class

[TMyConnectionOptions](#)

## Syntax

```
property Protocol: TMyProtocol default mpDefault;
```

## Remarks

Use the Protocol property to specify which protocol to use when connecting to server. To use these constants you should add MyClasses unit to uses clause.

© 1997-2012 Devart. All Rights Reserved.

### 17.15.1.9 MyAccess.TMyConnectionSSLOptions Class

This class allows setting up the behaviour of the TMyConnection class.

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

## Unit

[MyAccess](#)

## Syntax

```
TMyConnectionSSLOptions = class (TPersistent);
```

## Inheritance Hierarchy

TObject

**TMyConnectionSSLOptions**

© 1997-2012 Devart. All Rights Reserved.

[TMyConnectionSSLOptions](#) class overview.

## Properties

| Name                        | Description  |
|-----------------------------|--|
| <a href="#">CACert</a>      | Holds the pathname to the certificate authority file.        |
| <a href="#">Cert</a>        | Holds the pathname to the certificate file.                  |
| <a href="#">ChipherList</a> | Holds the list of allowed ciphers to use for SSL encryption. |
| <a href="#">Key</a>         | Holds the pathname to the key file.                          |

---

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyConnectionSSLOptions** class.

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

## Published

| Name                        | Description  |
|-----------------------------|--|
| <a href="#">CACert</a>      | Holds the pathname to the certificate authority file.        |
| <a href="#">Cert</a>        | Holds the pathname to the certificate file.                  |
| <a href="#">ChipherList</a> | Holds the list of allowed ciphers to use for SSL encryption. |
| <a href="#">Key</a>         | Holds the pathname to the key file.                          |

---

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Holds the pathname to the certificate authority file.

## Class

[TMyConnectionSSLOptions](#)

## Syntax

```
property CACert: string;
```

## Remarks

CACert is the pathname to the certificate authority file.

---

© 1997-2012 Devart. All Rights Reserved.

Holds the pathname to the certificate file.

## Class

[TMyConnectionSSLOptions](#)

## Syntax

```
property Cert: string;
```

## Remarks

Cert is the pathname to the certificate file.

© 1997-2012 Devart. All Rights Reserved.

Holds the list of allowed ciphers to use for SSL encryption.

### Class

[TMyConnectionSSLOptions](#)

### Syntax

```
property ChipherList: string;
```

### Remarks

ChipherList is the list of allowed ciphers to use for SSL encryption.

© 1997-2012 Devart. All Rights Reserved.

Holds the pathname to the key file.

### Class

[TMyConnectionSSLOptions](#)

### Syntax

```
property Key: string;
```

### Remarks

Key is the pathname to the key file.

© 1997-2012 Devart. All Rights Reserved.

#### 17.15.1.10 MyAccess.TMyDataSetOptions Class

This class allows setting up the behaviour of the TMyDataSet class.

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

### Unit

[MyAccess](#)

### Syntax

```
TMyDataSetOptions = class(TDADatasetOptions);
```

### Inheritance Hierarchy

TObject

[TDADatasetOptions](#)

**TMyDataSetOptions**

© 1997-2012 Devart. All Rights Reserved.

[TMyDataSetOptions](#) class overview.

### Properties

| Name                                | Description   |
|-------------------------------------|---|
| <a href="#">AutoPrepare</a>         | Used to execute automatic <a href="#">TCustomDADataset.Prepare</a> on a query execution.  |
| <a href="#">AutoRefresh</a>         | Used to automatically refresh dataset every AutoRefreshInterval seconds.  |
| <a href="#">AutoRefreshInterval</a> | Used to define in what time interval in seconds the Refresh or <a href="#">TCustomMyDataSet.RefreshQuick</a> method of a DataSet is called. |

|   |  |
|---|--|
| <a href="#">BinaryAsString</a>  | Used to specify a method of representation of the BINARY and VARBINARY fields.   |
| <a href="#">CacheCalcFields</a> (inherited from <a href="#">TDADatasetOptions</a> )   | Used to enable caching of the TField.Calculated and TField.Lookup fields.  |
| <a href="#">CheckRowVersion</a>   | Used to determine whether a dataset checks for rows modifications made by another user on automatic generation of SQL statement for update or delete data.                                   |
| <a href="#">CreateConnection</a>  | Used to specify if an additional connection to a server should be established to execute an additional query in the <a href="#">TCustomMyDataSet</a> . <a href="#">FetchAll</a> =False mode. |
| <a href="#">DefaultValues</a>   | Used to fill the DefaultExpression property of TField objects with appropriate value.  |
| <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="#">EnableBoolean</a>   | Used to specify the method of representation of the TINYINT(1) fields.   |
| <a href="#">FieldsAsString</a>  | Used to store all non-BLOB fields as string (native MySQL format).   |
| <a href="#">FieldsOrigin</a>  | Used to fill the Origin property of TField objects with appropriate value.   |
| <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 specify the fields to include in automatically generated SQL statement when calling the <a href="#">TCustomDADataset.RefreshRecord</a> method. Default value is false.               |
| <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="#">NullForZeroDate</a>   | Used for MySQL server to represent the value for for datetime fields with invalid values as Null or '0001-01-01' ('0100-01-01' for CLR).   |
| <a href="#">NumberRange</a>   | Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.   |
| <a href="#">QueryRecCount</a>   | Used for TCustomDADataset to perform additional query to get record count for this SELECT so the RecordCount property reflects the actual number of records.                                 |



|   |   |
|---|---|
| <a href="#">QuoteNames</a>  | Used for TCustomMyDataSet to quote all field names in autogenerated SQL statements.   |
| <a href="#">RemoveOnRefresh</a>   | Used for dataset to remove a record locally if the RefreshRecord procedure can't find necessary record on the server.                               |
| <a href="#">RequiredFields</a>  | Used for TCustomDADataset to set the Required property of TField objects for NOT NULL fields.   |
| <a href="#">ReturnParams</a>  | Used to return the new value of the fields to dataset after insert or update.   |
| <a href="#">SetFieldsReadOnly</a>   | Used to specify whether fields not belonging to the current updating table get read-only attribute.   |
| <a href="#">StrictUpdate</a>  | Used for TCustomDADataset to raise an exception when the number of updated or deleted records does not equal 1.                                     |
| <a href="#">TrimFixedChar</a>   | Used to specify whether to discard all trailing spaces in string fields of the 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. |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyDataSetOptions** class.

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

## Public

| Name  | Description   |
|---|---|
| <a href="#">CacheCalcFields</a> (inherited from <a href="#">TDADatasetOptions</a> )   | Used to enable caching of the TField.Calculated and TField.Lookup fields.   |
| <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="#">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="#">UpdateAllFields</a> (inherited from <a href="#">TDADatasetOptions</a> )   | 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.

## Published

| Name                                | Description  |
|-------------------------------------|--|
| <a href="#">AutoPrepare</a>         | Used to execute automatic <a href="#">TCustomDADataset.Prepare</a> on a query execution.   |
| <a href="#">AutoRefresh</a>         | Used to automatically refresh dataset every <a href="#">AutoRefreshInterval</a> seconds.   |
| <a href="#">AutoRefreshInterval</a> | Used to define in what time interval in seconds the Refresh or <a href="#">TCustomMyDataSet.RefreshQuick</a> method of a DataSet is called.                                    |
| <a href="#">BinaryAsString</a>      | Used to specify a method of representation of the BINARY and VARBINARY fields.   |
| <a href="#">CheckRowVersion</a>     | Used to determine whether a dataset checks for rows modifications made by another user on automatic generation of SQL statement for update or delete data.                     |
| <a href="#">CreateConnection</a>    | Used to specify if an additional connection to a server should be established to execute an additional query in the <a href="#">TCustomMyDataSet.FetchAll=False</a> mode.      |
| <a href="#">DefaultValues</a>       | Used to fill the DefaultExpression property of TField objects with appropriate value.  |
| <a href="#">EnableBoolean</a>       | Used to specify the method of representation of the TINYINT(1) fields.   |
| <a href="#">FieldsAsString</a>      | Used to store all non-BLOB fields as string (native MySQL format).   |
| <a href="#">FieldsOrigin</a>        | Used to fill the Origin property of TField objects with appropriate value.   |
| <a href="#">FullRefresh</a>         | Used to specify the fields to include in automatically generated SQL statement when calling the <a href="#">TCustomDADataset.RefreshRecord</a> method. Default value is false. |
| <a href="#">NullForZeroDate</a>     | Used for MySQL server to represent the value for for datetime fields with invalid values as Null or '0001-01-01' ('0100-01-01' for CLR).                                       |
| <a href="#">NumberRange</a>         | Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.   |
| <a href="#">QueryRecCount</a>       | Used for TCustomDADataset to perform additional query to get record count for this SELECT so the RecordCount property reflects the actual number of records.                   |

[QuoteNames](#)

Used for TCustomMyDataSet to quote all field names in autogenerated SQL statements.

[RemoveOnRefresh](#)

Used for dataset to remove a record locally if the RefreshRecord procedure can't find necessary record on the server.

[RequiredFields](#)

Used for TCustomDADataset to set the Required property of TField objects for NOT NULL fields.

[ReturnParams](#)

Used to return the new value of the fields to dataset after insert or update.

[SetFieldsReadOnly](#)

Used to specify whether fields not belonging to the current updating table get read-only attribute.

[StrictUpdate](#)

Used for TCustomDADataset to raise an exception when the number of updated or deleted records does not equal 1.

[TrimFixedChar](#)

Used to specify whether to discard all trailing spaces in string fields of the dataset.

**See Also**

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

© 1997-2012 Devart. All Rights Reserved.

Used to execute automatic [TCustomDADataset.Prepare](#) on a query execution.

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

**property** AutoPrepare: boolean;

**Remarks**

Set the AutoPrepare property to execute automatic [TCustomDADataset.Prepare](#) on a query execution. Makes sense for cases when a query will be executed several times, for example, in Master/Detail relationships.

© 1997-2012 Devart. All Rights Reserved.

Used to automatically refresh dataset every AutoRefreshInterval seconds.

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

**property** AutoRefresh: boolean **default** False;

**Remarks**

If True, dataset will be automatically refreshed every AutoRefreshInterval seconds. If dataset has at least one key field and a TIMESTAMP field, the [TCustomMyDataSet.RefreshQuick](#) method will be executed, otherwise the Refresh method will be executed. This option is only available for Windows.

© 1997-2012 Devart. All Rights Reserved.

Used to define in what time interval in seconds the Refresh or [TCustomMyDataSet.RefreshQuick](#) method of a DataSet is called.

**Class**

[TMyDataSetOptions](#)

**Syntax**

```
property AutoRefreshInterval: integer default 60;
```

**Remarks**

Use the AutoRefreshInterval property to define in what time interval in seconds the Refresh or [TCustomMyDataSet.RefreshQuick](#) method of a DataSet is called. This option is only available for Windows.

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify a method of representation of the BINARY and VARBINARY fields.

**Class**

[TMyDataSetOptions](#)

**Syntax**

```
property BinaryAsString: boolean default True;
```

**Remarks**

Use the BinaryAsString property to specify a method of representation of the BINARY and VARBINARY fields. If set to True, these fields will be represented as string fields; otherwise, as TBytesField and TVarBytesField correspondingly. If the binary fields should not be processed as strings then set this property to False. The default value is True.

---

© 1997-2012 Devart. All Rights Reserved.

Used to determine whether a dataset checks for rows modifications made by another user on automatic generation of SQL statement for update or delete data.

**Class**

[TMyDataSetOptions](#)

**Syntax**

```
property CheckRowVersion: boolean default False;
```

**Remarks**

Use the CheckRowVersion property to determine whether a dataset checks for rows modifications made by another user on automatic generation of SQL statement for update or delete data. If CheckRowVersion is True and DataSet has timestamp field when only this field is added into WHERE clause of generated SQL statement. If CheckRowVersion is True, but there is no TIMESTAMP field, then all nonblob fields will be added to WHERE clause. The default value is False.

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify if an additional connection to a server should be established to execute an additional query in the [TCustomMyDataSet.FetchAll](#)=False mode.

**Class**

[TMyDataSetOptions](#)

**Syntax**

```
property CreateConnection: boolean default True;
```

**Remarks**

Use the CreateConnection property to specify if an additional connection to a server should be established to execute an additional query in the [TCustomMyDataSet.FetchAll=False](#) mode. If a DataSet is opened in [TCustomMyDataSet.FetchAll=False](#), the current connection is blocked until all records have been fetched. If this option is set to True, an additional connection is created to fetch data to avoid blocking of the current connection.

---

© 1997-2012 Devart. All Rights Reserved.

Used to fill the DefaultExpression property of TField objects with appropriate value.

### Class

[TMyDataSetOptions](#)

### Syntax

**property** DefaultValues: boolean;

### Remarks

If True, TCustomMyDataSet fills the DefaultExpression property of TField objects with appropriate value.

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify the method of representation of the TINYINT(1) fields.

### Class

[TMyDataSetOptions](#)

### Syntax

**property** EnableBoolean: boolean **default** True;

### Remarks

Use the EnableBoolean property to specify the method of representation of the TINYINT(1) fields. If set to True, these fields will be represented as TBooleanField; otherwise, as TSmallintField. The default value is True.

---

© 1997-2012 Devart. All Rights Reserved.

Used to store all non-BLOB fields as string (native MySQL format).

### Class

[TMyDataSetOptions](#)

### Syntax

**property** FieldsAsString: boolean **default** False;

### Remarks

All non-BLOB fields are stored as string (native MySQL format). The default value is False.

---

© 1997-2012 Devart. All Rights Reserved.

Used to fill the Origin property of TField objects with appropriate value.

### Class

[TMyDataSetOptions](#)

### Syntax

**property** FieldsOrigin: boolean **default** True;

### Remarks

If True, TCustomMyDataSet fills the Origin property of TField objects with appropriate value.

© 1997-2012 Devart. All Rights Reserved.

Used to specify the fields to include in automatically generated SQL statement when calling the [TCustomDADataset.RefreshRecord](#) method. Default value is false.

### Class

[TMyDataSetOptions](#)

### Syntax

```
property FullRefresh: boolean;
```

### Remarks

Use the FullRefresh property to specify what fields to include in automatically generated SQL statement when calling the [TCustomDADataset.RefreshRecord](#) method. If the FullRefresh property is True, all the fields from query are included into a SQL statement to refresh single record. If FullRefresh is False, only fields from [TMyQuery.UpdatingTable](#) are included. The default value is False.

---

© 1997-2012 Devart. All Rights Reserved.

Used for MySQL server to represent the value for for datetime fields with invalid values as Null or '0001-01-01' ('0100-01-01' for CLR).

### Class

[TMyDataSetOptions](#)

### Syntax

```
property NullForZeroDate: boolean default True;
```

### Remarks

For datetime fields with invalid values, for example '2002-12-32', MySQL returns on fetch the '0000-00-00' value. According to the NullForZeroDate option this value will be represented as Null or '0001-01-01' ('0100-01-01' for CLR). The default value is True.

---

© 1997-2012 Devart. All Rights Reserved.

Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.

### Class

[TMyDataSetOptions](#)

### Syntax

```
property NumberRange: boolean;
```

### Remarks

Use the NumberRange property to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values. The default value is False.

---

© 1997-2012 Devart. All Rights Reserved.

Used for TCustomDADataset to perform additional query to get record count for this SELECT so the RecordCount property reflects the actual number of records.

### Class

[TMyDataSetOptions](#)

### Syntax

```
property QueryRecCount: boolean;
```

### Remarks

If True, and the [TCustomMyDataSet.FetchAll](#) property is False, TCustomDADataset performs additional query to get record count for this SELECT so the RecordCount property reflects the actual number of records. Does not have any effect if the [TCustomMyDataSet.FetchAll](#) property is True.

© 1997-2012 Devart. All Rights Reserved.

Used for TCustomMyDataSet to quote all field names in autogenerated SQL statements.

### Class

[TMyDataSetOptions](#)

### Syntax

**property** QuoteNames: boolean;

### Remarks

If True, TCustomMyDataSet quotes all field names in autogenerated SQL statements such as update SQL.

© 1997-2012 Devart. All Rights Reserved.

Used for dataset to remove a record locally if the RefreshRecord procedure can't find necessary record on the server.

### Class

[TMyDataSetOptions](#)

### Syntax

**property** RemoveOnRefresh: boolean;

### 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 its key value. 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.

© 1997-2012 Devart. All Rights Reserved.

Used for TCustomDADataset to set the Required property of TField objects for NOT NULL fields.

### Class

[TMyDataSetOptions](#)

### Syntax

**property** RequiredFields: boolean **default** False;

### Remarks

If True, TCustomDADataset sets the Required property of TField objects for NOT NULL fields. It is useful when table has a trigger that updates NOT NULL fields. The default value is False.

© 1997-2012 Devart. All Rights Reserved.

Used to return the new value of the fields to dataset after insert or update.

### Class

[TMyDataSetOptions](#)

### Syntax

**property** ReturnParams: boolean;

### Remarks

Use the ReturnParams property to return the new value of the fields to dataset after insert or update. Actual value of field after insert or update may be different from the value stored in local memory if the table has a trigger. When ReturnParams is True, OUT parameters of SQLInsert and SQLUpdate statements is assigned to corresponding fields. The default value is False.

© 1997-2012 Devart. All Rights Reserved.

Used to specify whether fields not belonging to the current updating table get read-only attribute.

### Class

[TMyDataSetOptions](#)

### Syntax

```
property SetFieldsReadOnly: boolean default False;
```

### Remarks

Use the SetFieldsReadOnly property to specify whether fields not belonging to the current updating table get read-only attribute. The default value is False.

© 1997-2012 Devart. All Rights Reserved.

Used for TCustomDADataset to raise an exception when the number of updated or deleted records does not equal 1.

### Class

[TMyDataSetOptions](#)

### Syntax

```
property StrictUpdate: boolean;
```

### Remarks

TCustomDADataset raises an exception when the number of updated or deleted records does 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 a non-SQL block. The default value is True.

© 1997-2012 Devart. All Rights Reserved.

Used to specify whether to discard all trailing spaces in string fields of the dataset.

### Class

[TMyDataSetOptions](#)

### Syntax

```
property TrimFixedChar: boolean stored False;
```

### Remarks

Use the TrimFixedChar property to specify whether to discard all trailing spaces in string fields of the dataset. The default value is True.

© 1997-2012 Devart. All Rights Reserved.

#### 17.15.1.11 MyAccess.TMyDataSource Class

TMyDataSource provides an interface between a MyDAC dataset components and data-aware controls on a form.

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

### Unit

[MyAccess](#)



## Syntax

```
TMyDataSource = class (TCRDataSource) ;
```

## Remarks

TMyDataSource provides an interface between a MyDAC dataset components and data-aware controls on a form.

TMyDataSource 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

TObject

[TCRDataSource](#)

**TMyDataSource**

© 1997-2012 Devart. All Rights Reserved.

[TMyDataSource](#) class overview.

© 1997-2012 Devart. All Rights Reserved.

### 17.15.1.12 MyAccess.TMyEncryptor Class

The class that performs encrypting and decrypting of data.

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

## Unit

[MyAccess](#)

## Syntax

```
TMyEncryptor = class (TCREncryptor) ;
```

## Inheritance Hierarchy

TObject

[TCREncryptor](#)

**TMyEncryptor**

© 1997-2012 Devart. All Rights Reserved.

[TMyEncryptor](#) 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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.15.1.13 MyAccess.TMyMetaData Class

A component for obtaining metainformation about database objects from the server.  
For a list of all members of this type, see [TMyMetaData](#) members.

### Unit

[MyAccess](#)

### Syntax

```
TMyMetaData = class (TDAMetaData) ;
```

### Remarks

The TMyMetaData component is used to obtain metainformation from the server about objects in the database, such as tables, table columns, stored procedures, etc. TMyMetaData publishes properties of [TDAMetaData](#).

### Inheritance Hierarchy

```

TObject
  TMemDataSet
    TDAMetaData
      TMyMetaData
  
```

### See Also

- [TCustomDADataset.Debug](#)
- [TCustomDASQL.Debug](#)
- [DBMonitor](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyMetaData](#) 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="#">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 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="#">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.                     |

## 17.15.1.14 MyAccess.TMyQuery 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 [TMyQuery](#) members.

## Unit

[MyAccess](#)

## Syntax

```
TMyQuery = class (TCustomMyDataSet) ;
```

## Remarks

TMyQuery is a direct descendant of the [TCustomMyDataSet](#) component. It publishes most of its inherited properties and events so that they can be manipulated at design-time.

Use TMyQuery to perform fetching, insertion, deletion and update of record by dynamically generated SQL statements. TMyQuery 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, TMyQuery will retrieve primary keys for UpdatingTable from metadata. TMyQuery 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

TObject

[TMemDataSet](#)

[TCustomDADataset](#)

[TCustomMyDataSet](#)

**TMyQuery**

## See Also

- Query demo project
- [Updating Data with MyDAC Dataset Components](#)
- [Master/Detail Relationships](#)
- [TMyStoredProc](#)
- [TMyTable](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyQuery](#) class overview.

## Properties

| Name  | Description   |
|---|---|
| <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="#">CommandTimeout</a> (inherited from <a href="#">TCustomMyDataSet</a> ) | Used to specify the amount of time to attempt execution of a command.                         |

|   |  |
|---|--|
| <a href="#"><u>Connection</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )   | Used to specify a connection object that will be used to connect to a data store.  |
| <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>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>FetchAll</u></a>   | Defines whether to request all records of the query from database server when the dataset is being opened.   |
| <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>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>InsertId</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )     | Returns the ID generated for an AUTO INCREMENT column by the previous query.   |
| <a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )      | Used to check whether 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>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>   | 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. |

[MasterSource](#) (inherited from [TCustomDADataset](#))

[Options](#) (inherited from [TCustomMyDataSet](#))

[ParamCheck](#) (inherited from [TCustomDADataset](#))

[ParamCount](#) (inherited from [TCustomDADataset](#))

[Params](#) (inherited from [TCustomDADataset](#))

[Prepared](#) (inherited from [TMemDataSet](#))

[ReadOnly](#) (inherited from [TCustomDADataset](#))

[RefreshOptions](#) (inherited from [TCustomDADataset](#))

[RowsAffected](#) (inherited from [TCustomDADataset](#))

[SQL](#) (inherited from [TCustomDADataset](#))

[SQLDelete](#) (inherited from [TCustomDADataset](#))

[SQLInsert](#) (inherited from [TCustomDADataset](#))

[SQLLock](#) (inherited from [TCustomDADataset](#))

[SQLRefresh](#) (inherited from [TCustomDADataset](#))

[SQLUpdate](#) (inherited from [TCustomDADataset](#))

[UniDirectional](#) (inherited from [TCustomDADataset](#))

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

[UpdatesPending](#) (inherited from [TMemDataSet](#))

[UpdatingTable](#)

Used to specify the data source component which binds current dataset to the master one.

Specifies the behaviour of TCustomMyDataSet object.

Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.

Used to indicate how many parameters are there in the Params property.

Used to view and set parameter names, values, and data types dynamically.

Determines whether a query is prepared for execution or not.

Used to prevent users from updating, inserting, or deleting data in the dataset.

Used to indicate when the editing record is refreshed.

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

Used to provide a SQL statement that a query component executes when its Open method is called.

Used to specify a SQL statement that will be used when applying a deletion to a record.

Used to specify the SQL statement that will be used when applying an insertion to a dataset.

Used to specify a SQL statement that will be used to perform a record lock.

Used to specify a SQL statement that will be used to refresh current record by calling the [TCustomDADataset.RefreshRecord](#) procedure.

Used to specify a SQL statement that will be used when applying an update to a dataset.

Used if an application does not need bidirectional access to records in the result set.

Used to indicate the update status for the current record when cached updates are enabled.

Used to check the status of the cached updates buffer.

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="#"><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>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>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>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>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>TCustomDADataset</u></a> )         | Determines if a parameter with the specified name exists in a dataset.   |
| <a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )                | Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself 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>GetFieldEnum</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Retrieve the list of acceptable values for a specified field given by the FieldName parameter.   |
| <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.   |

[GetOrderBy](#) (inherited from [TCustomDADataset](#))

[GotoCurrent](#) (inherited from [TCustomDADataset](#))

[Locate](#) (inherited from [TMemDataSet](#))

[LocateEx](#) (inherited from [TMemDataSet](#))

[Lock](#) (inherited from [TCustomMyDataSet](#))

[LockTable](#) (inherited from [TCustomMyDataSet](#))

[MacroByName](#) (inherited from [TCustomDADataset](#))

[ParamByName](#) (inherited from [TCustomDADataset](#))

[Prepare](#) (inherited from [TCustomDADataset](#))

[RefreshQuick](#) (inherited from [TCustomMyDataSet](#))

[RefreshRecord](#) (inherited from [TCustomDADataset](#))

[RestoreSQL](#) (inherited from [TCustomDADataset](#))

[RestoreUpdates](#) (inherited from [TMemDataSet](#))

[Resync](#) (inherited from [TCustomDADataset](#))

[RevertRecord](#) (inherited from [TMemDataSet](#))

[SaveSQL](#) (inherited from [TCustomDADataset](#))

[SaveToXML](#) (inherited from [TMemDataSet](#))

[SetOrderBy](#) (inherited from [TCustomDADataset](#))

[SQLSaved](#) (inherited from [TCustomDADataset](#))

[UnLock](#) (inherited from [TCustomDADataset](#))

[UnLockTable](#) (inherited from [TCustomMyDataSet](#))

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

Retrieves an ORDER BY clause from a SQL statement.

Sets the current record in this dataset similar to the current record in another dataset.

Overloaded. Searches a dataset for a specific record and positions the cursor on it.

Overloaded. Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet.

Overloaded. Locks the current record for the current connection.

Locks table for the current connection.

Finds a Macro with the name passed in Name.

Sets or uses parameter information for a specific parameter based on its name.

Allocates, opens, and parses cursor for a query.

Retrieves changes posted to the server by another clients on the client side quickly.

Actuali es field values for the current record.

Restores the SQL property modified by AddWhere and SetOrderBy.

Marks all records in the cache of updates as unapplied.

Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.

Cancels changes made to the current record when cached updates are enabled.

Saves the SQL property value to BaseSQL.

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Builds an ORDER BY clause of a SELECT statement.

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

Releases a record lock.

Releases a table locked by the [TCustomMyDataSet.LockTable](#) method.

Frees the resources allocated for a previously prepared query on the server and client sides.

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.                     |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyQuery** class.

For a complete list of the **TMyQuery** class members, see the [TMyQuery 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="#">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="#"><u>CommandTimeout</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )   | Used to specify the amount of time to attempt execution of a command.  |
| <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>TCustomMyDataSet</u></a> )       | Used to specify a connection object that will be used to connect to a data store.  |
| <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>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>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="#"><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>TCustomDADataset</u></a> )        | Determines if a parameter with the specified name exists in a dataset.   |

|  |  |
|--|--|
| <a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )                | Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself 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>GetFieldEnum</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Retrieve the list of acceptable values for a specified field given by the FieldName parameter.                                       |
| <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>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>InsertId</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )          | Returns the ID generated for an AUTO INCREMENT column by the previous query.   |
| <a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )           | Used to check whether 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>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>TCustomMyDataSet</u></a> )              | Overloaded. Locks the current record for the current connection.   |
| <a href="#"><u>LockTable</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )         | Locks table for the current connection.  |
| <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>TCustomMyDataSet</u></a> )        | Specifies the behaviour of TCustomMyDataSet object.  |
| <a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )    | Sets or uses 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>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>RefreshQuick</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )   | Retrieves changes posted to the server by another clients on the client side quickly.  |
| <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="#"><u>Resync</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )         | Resynchroni e the dataset with underlying physical data when making calls that may change the internal cursor position.  |
| <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>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>SaveSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></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="#">UnLockTable</a> (inherited from <a href="#">TCustomMyDataSet</a> )    | Releases a table locked by the <a href="#">TCustomMyDataSet.LockTable</a> method.  |
| <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.   |

## 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.                                 |

[UpdatingTable](#)

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**

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

© 1997-2012 Devart. All Rights Reserved.

Defines whether to request all records of the query from database server when the dataset is being opened.

**Class**

[TMyQuery](#)

**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.

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify what kind of lock will be performed when editing a record.

**Class**

[TMyQuery](#)

**Syntax**

```
property LockMode: TLockMode;
```

**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 ImNone.

**See Also**

- [TMyStoredProc.LockMode](#)
  - [TMyTable.LockMode](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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**

[TMyQuery](#)

**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 [TCustomMyDataSet.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. All fields from other than target table have their ReadOnly properties set to True (if [TCustomMyDataSet.Options](#) )

## Example

The first example specifies a query in which only one value 'Dept' for the UpdatingTable property is allowed.

The second example shows a query in which allowed values for UpdatingTable are 'Dept' and 'Emp'. By default the updating table will be the first used table, so 'DEPT' and all fields of DEPT will be editable. If however UpdatingTable is set to be 'EMP' all fields of EMP will be editable.

Example 1.

```
SELECT * FROM Dept
```

Example 2.

```
SELECT * FROM Dept, Emp
WHERE Dept.DeptNo = Emp.DeptNo
```

© 1997-2012 Devart. All Rights Reserved.

### 17.15.1.15 MyAccess.TMyStoredProc Class

A component for accessing and executing stored procedures and functions.

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

## Unit

[MyAccess](#)

## Syntax

```
TMyStoredProc = class (TCustomMyStoredProc) ;
```

## Remarks

Use TMyStoredProc 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 return parameters in the Params property.

## Inheritance Hierarchy

TObject

[TMemDataSet](#)

[TCustomDADataset](#)

[TCustomMyDataSet](#)

[TCustomMyStoredProc](#)

**TMyStoredProc**

## See Also

- Stored proc demo
- [TMyQuery](#)
- [TMyCommand](#)
- [Updating Data with MyDAC Dataset Components](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyStoredProc](#) class overview.

## Properties

| Name  | Description  |
|---|--|
| <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="#">CommandTimeout</a> (inherited from <a href="#">TCustomMyDataSet</a> ) | Used to specify the amount of time to attempt execution of a command.  |
| <a href="#">Connection</a> (inherited from <a href="#">TCustomMyDataSet</a> )     | Used to specify a connection object that will be used to connect to a data store.  |
| <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="#">Encryption</a> (inherited from <a href="#">TCustomDADataset</a> )     | Used to specify the options of the data encryption in a dataset.   |
| <a href="#">FetchAll</a> (inherited from <a href="#">TCustomMyDataSet</a> )       | Description is not available at the moment.  |
| <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="#">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="#">InsertId</a> (inherited from <a href="#">TCustomMyDataSet</a> )       | Returns the ID generated for an AUTO INCREMENT column by the previous query.   |
| <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="#">LockMode</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>TCustomMyDataSet</u></a> )           | Specifies the behaviour of TCustomMyDataSet 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>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>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>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>StoredProcName</u></a> (inherited from <a href="#"><u>TCustomMyStoredProc</u></a> ) | Used to specify the name of the stored procedure to call on the server.  |

[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.

[UpdatingTable](#)

Specifies 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="#">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="#">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> (inherited from <a href="#">TCustomMyStoredProc</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="#">TCustomDADataset</a> )          | Used to learn whether 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="#">TCustomDADataset</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> (inherited from <a href="#">TCustomDADataset</a> )       | Returns internal field types defined in the MemData and accompanying modules.   |
| <a href="#">GetFieldEnum</a> (inherited from <a href="#">TCustomMyDataSet</a> )      | Retrieve the list of acceptable values for a specified field given by the FieldName parameter.                                |
| <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="#">TCustomMyDataSet</a> )              | Overloaded. Locks the current record for the current connection.  |
| <a href="#">LockTable</a> (inherited from <a href="#">TCustomMyDataSet</a> )         | Locks table for the current connection.   |
| <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="#">TCustomDADataset</a> )       | Sets or uses 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="#">PrepareSQL</a> (inherited from <a href="#">TCustomMyStoredProc</a> )     | Builds a query for TCustomMyStoredProc based on the Params and StoredProcName properties, and assigns it to the SQL property. |
| <a href="#">RefreshQuick</a> (inherited from <a href="#">TCustomMyDataSet</a> )      | Retrieves changes posted to the server by another clients on the client side quickly.   |
| <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.       |

[RevertRecord](#) (inherited from [TMemDataSet](#))

[SaveSQL](#) (inherited from [TCustomDADataset](#))

[SaveToXML](#) (inherited from [TMemDataSet](#))

[SetOrderBy](#) (inherited from [TCustomDADataset](#))

[SQLSaved](#) (inherited from [TCustomDADataset](#))

[UnLock](#) (inherited from [TCustomDADataset](#))

[UnLockTable](#) (inherited from [TCustomMyDataSet](#))

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Cancels changes made to the current record when cached updates are enabled.

Saves the SQL property value to BaseSQL.

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Builds an ORDER BY clause of a SELECT statement.

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

Releases a record lock.

Releases a table locked by the [TCustomMyDataSet.LockTable](#) method.

Frees the resources allocated for a previously prepared query on the server and client sides.

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.

## 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.                     |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyStoredProc** class.

For a complete list of the **TMyStoredProc** class members, see the [TMyStoredProc 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>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>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>CommandTimeout</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Used to specify the amount of time to attempt execution of a command.  |
| <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>TCustomMyDataSet</u></a> )          | Used to specify a connection object that will be used to connect to a data store.  |
| <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>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>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> (inherited from <a href="#"><u>TCustomMyStoredProc</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>FetchAll</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )            | Description is not available at the moment.  |

|  |  |
|--|--|
| <a href="#"><u>Fetch</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="#"><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>TCustomDADataset</u></a> )         | Determines if a parameter with the specified name exists in a dataset.   |
| <a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )                | Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself 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>GetFieldEnum</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Retrieve the list of acceptable values for a specified field given by the FieldName parameter.   |
| <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>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>InsertId</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )          | Returns the ID generated for an AUTO INCREMENT column by the previous query.   |
| <a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></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="#">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="#">TCustomMyDataSet</a> )         | Overloaded. Locks the current record for the current connection.   |
| <a href="#">LockTable</a> (inherited from <a href="#">TCustomMyDataSet</a> )    | Locks table for the current connection.  |
| <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> (inherited from <a href="#">TCustomMyDataSet</a> )      | Specifies the behaviour of TCustomMyDataSet 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="#">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="#">PrepareSQL</a> (inherited from <a href="#">TCustomMyStoredProc</a> )  | Builds a query for TCustomMyStoredProc based on the Params and StoredProcName properties, and assigns it to the SQL property.                        |
| <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="#">RefreshQuick</a> (inherited from <a href="#">TCustomMyDataSet</a> )   | Retrieves changes posted to the server by another clients on the client side quickly.  |
| <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="#">StoredProcName</a> (inherited from <a href="#">TCustomMyStoredProc</a> ) | Used to specify the name of the stored procedure to call on the server.                          |
| <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="#">UnLockTable</a> (inherited from <a href="#">TCustomMyDataSet</a> )       | Releases a table locked by the <a href="#">TCustomMyDataSet.LockTable</a> method.                |
| <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.             |

## Published

| Name                          | Description  |
|-------------------------------|--|
| <a href="#">LockMode</a>      | Used to specify what kind of lock will be performed when editing a record.   |
| <a href="#">UpdatingTable</a> | Specifies 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

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

© 1997-2012 Devart. All Rights Reserved.

Used to specify what kind of lock will be performed when editing a record.

## Class

[TMyStoredProc](#)

## Syntax

**property** LockMode: [TLockMode](#);

## 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 ImNone.

## See Also

- [TMyQuery.LockMode](#)
- [TMyTable.LockMode](#)

© 1997-2012 Devart. All Rights Reserved.

Specifies 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

[TMyStoredProc](#)

## Syntax

```
property UpdatingTable: string;
```

## Remarks

the UpdatingTable property is used on Insert, Update, Delete or RefreshRecord (see also [TCustomMyDataSet.Options](#)) if appropriate SQL (SQLInsert, SQLUpdate or SQLDelete) is not provided. If UpdatingTable is not set then the first table used in query is assumed to be the target. If the query is addressed to the View then entire View is taken as a target for subsequent modifications. All fields from other than target table have their ReadOnly properties set to True (if [TCustomMyDataSet.Options](#) is True).

© 1997-2012 Devart. All Rights Reserved.

### 17.15.1.16 MyAccess.TMyTable 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 [TMyTable](#) members.

## Unit

[MyAccess](#)

## Syntax

```
TMyTable = class (TCustomMyTable) ;
```

## Remarks

The TMyTable component allows retrieving and updating data in a single table without writing SQL statements. Use TMyTable to access data in a table . Use the TableName property to specify table name. TMyTable 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

```
TObject
  TMemDataSet
    TCustomDADataset
      TCustomMyDataSet
        TCustomMyTable
          TMyTable
```

## See Also

- [Updating Data with MyDAC Dataset Components](#)
- [Master/Detail Relationships](#)
- [TCustomMyDataSet](#)
- [TMyQuery](#)
- [TCustomMyTable](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyTable](#) class overview.

## Properties

| Name  | Description  |
|---|--|
| <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="#">CommandTimeout</a> (inherited from <a href="#">TCustomMyDataSet</a> ) | Used to specify the amount of time to attempt execution of a command.  |
| <a href="#">Connection</a> (inherited from <a href="#">TCustomMyDataSet</a> )     | Used to specify a connection object that will be used to connect to a data store.  |
| <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="#">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="#">IndexDefs</a> (inherited from <a href="#">TCustomMyTable</a> )        | Contains information about the indexes for a table.  |
| <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="#">InsertId</a> (inherited from <a href="#">TCustomMyDataSet</a> )       | Returns the ID generated for an AUTO INCREMENT column by the previous query.   |
| <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="#">Limit</a> (inherited from <a href="#">TCustomMyTable</a> )            | Used to set the number of rows retrieved from the query.   |
| <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="#"><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>   | 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>Offset</u></a> (inherited from <a href="#"><u>TCustomMyTable</u></a> )           | Used to allow retrieving data from the server starting from the specified row.   |
| <a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomMyTable</u></a> )          | Specifies the behaviour of the <a href="#"><u>TMyTable</u></a> object.   |
| <a href="#"><u>OrderFields</u></a>  | Used to build ORDER BY clause of SQL statements.   |
| <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>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>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.  |

[SQLRefresh](#) (inherited from [TCustomDADataset](#))

Used to specify a SQL statement that will be used to refresh current record by calling the [TCustomDADataset.RefreshRecord](#) procedure.

[SQLUpdate](#) (inherited from [TCustomDADataset](#))

Used to specify a SQL statement that will be used when applying an update to a dataset.

[TableName](#)

Used to specify the name of the database table this component encapsulates.

[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.

## 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="#">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="#">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="#">EmptyTable</a> (inherited from <a href="#">TCustomMyTable</a> )         | Deletes all records from the database table specified by the TableName 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> (inherited from <a href="#">TCustomDADataset</a> )          | Used to learn whether 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.   |

[FindMacro](#) (inherited from [TCustomDADataset](#))

[FindNearest](#) (inherited from [TCustomDADataset](#))

[FindParam](#) (inherited from [TCustomDADataset](#))

[GetBlob](#) (inherited from [TMemDataSet](#))

[GetDataType](#) (inherited from [TCustomDADataset](#))

[GetFieldEnum](#) (inherited from [TCustomMyDataSet](#))

[GetFieldObject](#) (inherited from [TCustomDADataset](#))

[GetFieldPrecision](#) (inherited from [TCustomDADataset](#))

[GetFieldScale](#) (inherited from [TCustomDADataset](#))

[GetOrderBy](#) (inherited from [TCustomDADataset](#))

[GotoCurrent](#) (inherited from [TCustomDADataset](#))

[Locate](#) (inherited from [TMemDataSet](#))

[LocateEx](#) (inherited from [TMemDataSet](#))

[Lock](#) (inherited from [TCustomMyDataSet](#))

[LockTable](#) (inherited from [TCustomMyDataSet](#))

[MacroByName](#) (inherited from [TCustomDADataset](#))

[ParamByName](#) (inherited from [TCustomDADataset](#))

[Prepare](#) (inherited from [TCustomDADataset](#))

[RefreshQuick](#) (inherited from [TCustomMyDataSet](#))

[RefreshRecord](#) (inherited from [TCustomDADataset](#))

[RestoreSQL](#) (inherited from [TCustomDADataset](#))

[RestoreUpdates](#) (inherited from [TMemDataSet](#))

Indicates whether a specified macro exists in a dataset.

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.

Determines if a parameter with the specified name exists in a dataset.

Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.

Returns internal field types defined in the MemData and accompanying modules.

Retrieve the list of acceptable values for a specified field given by the FieldName parameter.

Returns a multireference shared object from field.

Retrieves the precision of a number field.

Retrieves the scale of a number field.

Retrieves an ORDER BY clause from a SQL statement.

Sets the current record in this dataset similar to the current record in another dataset.

Overloaded. Searches a dataset for a specific record and positions the cursor on it.

Overloaded. Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet.

Overloaded. Locks the current record for the current connection.

Locks table for the current connection.

Finds a Macro with the name passed in Name.

Sets or uses parameter information for a specific parameter based on its name.

Allocates, opens, and parses cursor for a query.

Retrieves changes posted to the server by another clients on the client side quickly.

Actuali es field values for the current record.

Restores the SQL property modified by AddWhere and SetOrderBy.

Marks all records in the cache of updates as unapplied.

[Resync](#) (inherited from [TCustomDADataset](#))

[RevertRecord](#) (inherited from [TMemDataSet](#))

[SaveSQL](#) (inherited from [TCustomDADataset](#))

[SaveToXML](#) (inherited from [TMemDataSet](#))

[SetOrderBy](#) (inherited from [TCustomDADataset](#))

[SQLSaved](#) (inherited from [TCustomDADataset](#))

[UnLock](#) (inherited from [TCustomDADataset](#))

[UnLockTable](#) (inherited from [TCustomMyDataSet](#))

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Resynchronizes the dataset with underlying physical data when making calls that may change the internal cursor position.

Cancels changes made to the current record when cached updates are enabled.

Saves the SQL property value to BaseSQL.

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Builds an ORDER BY clause of a SELECT statement.

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

Releases a record lock.

Releases a table locked by the [TCustomMyDataSet.LockTable](#) method.

Frees the resources allocated for a previously prepared query on the server and client sides.

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.

## 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.                     |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyTable** class.

For a complete list of the **TMyTable** class members, see the [TMyTable 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>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>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>CommandTimeout</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Used to specify the amount of time to attempt execution of a command.  |
| <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>TCustomMyDataSet</u></a> )          | Used to specify a connection object that will be used to connect to a data store.  |
| <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>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>EmptyTable</u></a> (inherited from <a href="#"><u>TCustomMyTable</u></a> )            | Deletes all records from the database table specified by the TableName property.   |
| <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="#"><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>TCustomDADataset</u></a> )         | Determines if a parameter with the specified name exists in a dataset.   |
| <a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )                | Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself 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>GetFieldEnum</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Retrieve the list of acceptable values for a specified field given by the FieldName parameter.   |
| <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>IndexDefs</u></a> (inherited from <a href="#"><u>TCustomMyTable</u></a> )           | Contains information about the indexes for a table.  |
| <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>InsertId</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )          | Returns the ID generated for an AUTO INCREMENT column by the previous query.   |

|   |  |
|---|--|
| <a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )      | Used to check whether 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>Limit</u></a> (inherited from <a href="#"><u>TCustomMyTable</u></a> )          | Used to set the number of rows retrieved from the query.   |
| <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>TCustomMyDataSet</u></a> )         | Overloaded. Locks the current record for the current connection.   |
| <a href="#"><u>LockTable</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )    | Locks table for the current connection.  |
| <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>Offset</u></a> (inherited from <a href="#"><u>TCustomMyTable</u></a> )         | Used to allow retrieving data from the server starting from the specified row.   |
| <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>TCustomMyTable</u></a> )        | Specifies the behaviour of the <a href="#"><u>TMyTable</u></a> object.   |
| <a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )  | Sets or uses 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="#">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="#">RefreshQuick</a> (inherited from <a href="#">TCustomMyDataSet</a> )   | Retrieves changes posted to the server by another clients on the client side quickly.  |
| <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. |

[SQLSaved](#) (inherited from [TCustomDADataset](#))

[SQLUpdate](#) (inherited from [TCustomDADataset](#))

[UniDirectional](#) (inherited from [TCustomDADataset](#))

[UnLock](#) (inherited from [TCustomDADataset](#))

[UnLockTable](#) (inherited from [TCustomMyDataSet](#))

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

[UpdatesPending](#) (inherited from [TMemDataSet](#))

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

Used to specify a SQL statement that will be used when applying an update to a dataset.

Used if an application does not need bidirectional access to records in the result set.

Releases a record lock.

Releases a table locked by the [TCustomMyDataSet.LockTable](#) method.

Frees the resources allocated for a previously prepared query on the server and client sides.

Used to indicate the update status for the current record when cached updates are enabled.

Reads the status of the latest call to the [ApplyUpdates](#) method while cached updates are enabled.

Used to check the status of the cached updates buffer.

Indicates the current update status for the dataset when cached updates are enabled.

## 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="#">OrderFields</a> | Used to build ORDER BY clause of SQL statements.   |
| <a href="#">TableName</a>   | Used to specify the name of the database table this component encapsulates.                                |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Defines whether to request all records of the query from database server when the dataset is being opened.

## Class

[TMyTable](#)

## 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.

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify what kind of lock will be performed when editing a record.

### Class

[TMyTable](#)

### Syntax

**property** LockMode: [TLockMode](#);

### 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 ImNone.

### See Also

- [TMyStoredProc.LockMode](#)
- [TMyQuery.LockMode](#)

---

© 1997-2012 Devart. All Rights Reserved.

Used to build ORDER BY clause of SQL statements.

### Class

[TMyTable](#)

### Syntax

**property** OrderFields: string;

### Remarks

TMyTable uses the OrderFields property to build ORDER BY clause of SQL statements. To set several field names to this property separate them with commas.  
TMyTable is reopened when OrderFields is being changed.

### See Also

- [TMyTable](#)

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify the name of the database table this component encapsulates.

### Class

[TMyTable](#)

### Syntax

**property** TableName: string;

### Remarks

Use the TableName property to specify the name of the database table this component encapsulates. If [TCustomDADDataSet.Connection](#) is assigned at design time, select a valid table name from the TableName

drop-down list in Object Inspector.

© 1997-2012 Devart. All Rights Reserved.

#### 17.15.1.17 MyAccess.TMyTableOptions Class

This class allows setting up the behaviour of the TMyTable class.  
For a list of all members of this type, see [TMyTableOptions](#) members.

### Unit

[MyAccess](#)

### Syntax

```
TMyTableOptions = class (TMyDataSetOptions);
```

### Inheritance Hierarchy

```

TObject
  TDADatasetOptions
    TMyDataSetOptions
      TMyTableOptions

```

© 1997-2012 Devart. All Rights Reserved.

[TMyTableOptions](#) class overview.

### Properties

| Name  | Description  |
|---|--|
| <a href="#">AutoPrepare</a> (inherited from <a href="#">TMyDataSetOptions</a> )         | Used to execute automatic <a href="#">TCustomDADataset.Prepare</a> on a query execution.   |
| <a href="#">AutoRefresh</a> (inherited from <a href="#">TMyDataSetOptions</a> )         | Used to automatically refresh dataset every AutoRefreshInterval seconds.   |
| <a href="#">AutoRefreshInterval</a> (inherited from <a href="#">TMyDataSetOptions</a> ) | Used to define in what time interval in seconds the Refresh or <a href="#">TCustomMyDataSet.RefreshQuick</a> method of a DataSet is called.                                |
| <a href="#">BinaryAsString</a> (inherited from <a href="#">TMyDataSetOptions</a> )      | Used to specify a method of representation of the BINARY and VARBINARY fields.   |
| <a href="#">CacheCalcFields</a> (inherited from <a href="#">TDADatasetOptions</a> )     | Used to enable caching of the TField.Calculated and TField.Lookup fields.  |
| <a href="#">CheckRowVersion</a> (inherited from <a href="#">TMyDataSetOptions</a> )     | Used to determine whether a dataset checks for rows modifications made by another user on automatic generation of SQL statement for update or delete data.                 |
| <a href="#">CreateConnection</a> (inherited from <a href="#">TMyDataSetOptions</a> )    | Used to specify if an additional connection to a server should be established to execute an additional query in the <a href="#">TCustomMyDataSet.FetchAll</a> =False mode. |
| <a href="#">DefaultValues</a> (inherited from <a href="#">TMyDataSetOptions</a> )       | Used to fill the DefaultExpression property of TField objects with appropriate value.  |
| <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="#">EnableBoolean</a> (inherited from <a href="#">TMyDataSetOptions</a> )     | Used to specify the method of representation of the TINYINT(1) fields.   |
| <a href="#">FieldsAsString</a> (inherited from <a href="#">TMyDataSetOptions</a> )    | Used to store all non-BLOB fields as string (native MySQL format).   |
| <a href="#">FieldsOrigin</a> (inherited from <a href="#">TMyDataSetOptions</a> )      | Used to fill the Origin property of TField objects with appropriate value.   |
| <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> (inherited from <a href="#">TMyDataSetOptions</a> )       | Used to specify the fields to include in automatically generated SQL statement when calling the <a href="#">TCustomDADataset.RefreshRecord</a> method. Default value is false. |
| <a href="#">HandlerIndex</a>  | Used to assign an index and a value that this index should satisfy.  |
| <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="#">NullForZeroDate</a> (inherited from <a href="#">TMyDataSetOptions</a> )   | Used for MySQL server to represent the value for for datetime fields with invalid values as Null or '0001-01-01' ('0100-01-01' for CLR).                                       |
| <a href="#">NumberRange</a> (inherited from <a href="#">TMyDataSetOptions</a> )       | Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.   |
| <a href="#">QueryRecCount</a> (inherited from <a href="#">TMyDataSetOptions</a> )     | Used for TCustomDADataset to perform additional query to get record count for this SELECT so the RecordCount property reflects the actual number of records.                   |
| <a href="#">QuoteNames</a> (inherited from <a href="#">TMyDataSetOptions</a> )        | Used for TCustomMyDataSet to quote all field names in autogenerated SQL statements.  |
| <a href="#">RemoveOnRefresh</a> (inherited from <a href="#">TMyDataSetOptions</a> )   | Used for dataset to remove a record locally if the RefreshRecord procedure can't find necessary record on the server.  |
| <a href="#">RequiredFields</a> (inherited from <a href="#">TMyDataSetOptions</a> )    | Used for TCustomDADataset to set the Required property of TField objects for NOT NULL fields.  |
| <a href="#">ReturnParams</a> (inherited from <a href="#">TMyDataSetOptions</a> )      | Used to return the new value of the fields to dataset after insert or update.  |
| <a href="#">SetFieldsReadOnly</a> (inherited from <a href="#">TMyDataSetOptions</a> ) | Used to specify whether fields not belonging to the current updating table get read-only attribute.  |
| <a href="#">StrictUpdate</a> (inherited from <a href="#">TMyDataSetOptions</a> )      | Used for TCustomDADataset to raise an exception when the number of updated or deleted records does not equal 1.  |
| <a href="#">TrimFixedChar</a> (inherited from <a href="#">TMyDataSetOptions</a> )     | Used to specify whether to discard all trailing spaces in string fields of the 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. |
| <a href="#">UseHandler</a>  | Used for the HANDLER statement to be used instead of the SELECT statement.  |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyTableOptions** class.

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

## Public

| Name  | Description   |
|---|---|
| <a href="#">CacheCalcFields</a> (inherited from <a href="#">TDADatasetOptions</a> )   | Used to enable caching of the TField.Calculated and TField.Lookup fields.   |
| <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="#">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="#">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="#">AutoPrepare</a> (inherited from <a href="#">TMyDataSetOptions</a> )         | Used to execute automatic <a href="#">TCustomDADataset.Prepare</a> on a query execution.  |
| <a href="#">AutoRefresh</a> (inherited from <a href="#">TMyDataSetOptions</a> )         | Used to automatically refresh dataset every AutoRefreshInterval seconds.  |
| <a href="#">AutoRefreshInterval</a> (inherited from <a href="#">TMyDataSetOptions</a> ) | Used to define in what time interval in seconds the Refresh or <a href="#">TCustomMyDataSet.RefreshQuick</a> method of a DataSet is called. |
| <a href="#">BinaryAsString</a> (inherited from <a href="#">TMyDataSetOptions</a> )      | Used to specify a method of representation of the BINARY and VARBINARY fields.  |



|   |  |
|---|--|
| <a href="#">CheckRowVersion</a> (inherited from <a href="#">TMyDataSetOptions</a> )   | Used to determine whether a dataset checks for rows modifications made by another user on automatic generation of SQL statement for update or delete data.                     |
| <a href="#">CreateConnection</a> (inherited from <a href="#">TMyDataSetOptions</a> )  | Used to specify if an additional connection to a server should be established to execute an additional query in the <a href="#">TCustomMyDataSet.FetchAll</a> =False mode.     |
| <a href="#">DefaultValues</a> (inherited from <a href="#">TMyDataSetOptions</a> )     | Used to fill the DefaultExpression property of TField objects with appropriate value.  |
| <a href="#">EnableBoolean</a> (inherited from <a href="#">TMyDataSetOptions</a> )     | Used to specify the method of representation of the TINYINT(1) fields.   |
| <a href="#">FieldsAsString</a> (inherited from <a href="#">TMyDataSetOptions</a> )    | Used to store all non-BLOB fields as string (native MySQL format).   |
| <a href="#">FieldsOrigin</a> (inherited from <a href="#">TMyDataSetOptions</a> )      | Used to fill the Origin property of TField objects with appropriate value.   |
| <a href="#">FullRefresh</a> (inherited from <a href="#">TMyDataSetOptions</a> )       | Used to specify the fields to include in automatically generated SQL statement when calling the <a href="#">TCustomDADataset.RefreshRecord</a> method. Default value is false. |
| <a href="#">HandlerIndex</a>  | Used to assign an index and a value that this index should satisfy.  |
| <a href="#">NullForZeroDate</a> (inherited from <a href="#">TMyDataSetOptions</a> )   | Used for MySQL server to represent the value for for datetime fields with invalid values as Null or '0001-01-01' ('0100-01-01' for CLR).                                       |
| <a href="#">NumberRange</a> (inherited from <a href="#">TMyDataSetOptions</a> )       | Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.   |
| <a href="#">QueryRecCount</a> (inherited from <a href="#">TMyDataSetOptions</a> )     | Used for TCustomDADataset to perform additional query to get record count for this SELECT so the RecordCount property reflects the actual number of records.                   |
| <a href="#">QuoteNames</a> (inherited from <a href="#">TMyDataSetOptions</a> )        | Used for TCustomMyDataSet to quote all field names in autogenerated SQL statements.  |
| <a href="#">RemoveOnRefresh</a> (inherited from <a href="#">TMyDataSetOptions</a> )   | Used for dataset to remove a record locally if the RefreshRecord procedure can't find necessary record on the server.  |
| <a href="#">RequiredFields</a> (inherited from <a href="#">TMyDataSetOptions</a> )    | Used for TCustomDADataset to set the Required property of TField objects for NOT NULL fields.  |
| <a href="#">ReturnParams</a> (inherited from <a href="#">TMyDataSetOptions</a> )      | Used to return the new value of the fields to dataset after insert or update.  |
| <a href="#">SetFieldsReadOnly</a> (inherited from <a href="#">TMyDataSetOptions</a> ) | Used to specify whether fields not belonging to the current updating table get read-only attribute.  |
| <a href="#">StrictUpdate</a> (inherited from <a href="#">TMyDataSetOptions</a> )      | Used for TCustomDADataset to raise an exception when the number of updated or deleted records does not equal 1.  |

[TrimFixedChar](#) (inherited from [TMyDataSetOptions](#))

Used to specify whether to discard all trailing spaces in string fields of the dataset.

[UseHandler](#)

Used for the HANDLER statement to be used instead of the SELECT statement.

### See Also

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

© 1997-2012 Devart. All Rights Reserved.

Used to assign an index and a value that this index should satisfy.

### Class

[TMyTableOptions](#)

### Syntax

```
property HandlerIndex: string;
```

### Remarks

Use the HandlerIndex property to assign an index and a value that this index should satisfy.

© 1997-2012 Devart. All Rights Reserved.

Used for the HANDLER statement to be used instead of the SELECT statement.

### Class

[TMyTableOptions](#)

### Syntax

```
property UseHandler: boolean default False;
```

### Remarks

If this option is enabled, the HANDLER statement is used instead of the SELECT statement.

From the MySQL Reference Manual:

*"There are several reasons to use the HANDLER interface instead of normal SELECT statements:*

*HANDLER is faster than SELECT:*

*A designated storage engine handler object is allocated for the HANDLER ... OPEN. The object is reused for subsequent HANDLER statements for that table; it need not be reinitialized for each one.*

*There is less parsing involved.*

*There is no optimizer or query-checking overhead.*

*The table does not have to be locked between two handler requests.*

*The handler interface does not have to provide a consistent look of the data (for example, dirty reads are allowed), so the storage engine can use optimizations that SELECT does not normally allow.*

*For applications that use a low-level ISAM-like interface, HANDLER makes it much easier to port them to MySQL.*

*HANDLER enables you to traverse a database in a manner that is difficult (or even impossible) to accomplish with SELECT. The HANDLER interface is a more natural way to look at data when working with applications that provide an interactive user interface to the database."*

The FilterSQL property is used to assign the WHERE condition. To return a specific number of rows, assign the Limit property.

If the Limit property equals -1 and the UseHandler option is set to True, requested records count equals to MaxInt.

The default value of the UseHandler option is False.

© 1997-2012 Devart. All Rights Reserved.

## 17.15.1.18 MyAccess.TMyTransaction Class

A component for managing transactions.

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

**Unit**

[MyAccess](#)

**Syntax**

```
TMyTransaction = class (TDATransaction) ;
```

**Remarks**

The TMyTransaction component is used to manage transactions in your application.

**Inheritance Hierarchy**

TObject

[TDATransaction](#)

**TMyTransaction**

© 1997-2012 Devart. All Rights Reserved.

[TMyTransaction](#) class overview.

**Properties**

| Name  | Description  |
|---|--|
| <a href="#">Active</a> (inherited from <a href="#">TDATransaction</a> )             | Used to determine if the transaction is active.  |
| <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. |

**Methods**

| Name  | Description  |
|---|--|
| <a href="#">Commit</a> (inherited from <a href="#">TDATransaction</a> )           | Commits the current 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.  |

**Events**

| Name   | Description   |
|--|---|
| <a href="#">OnError</a> (inherited from <a href="#">TDATransaction</a> ) | Used to process errors that occur during executing a transaction. |

© 1997-2012 Devart. All Rights Reserved.

## 17.15.1.19 MyAccess.TMyUpdateSQL Class

A component for tuning update operations for the DataSet component.

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

**Unit**

[MyAccess](#)

**Syntax**

```
TMyUpdateSQL = class (TCustomDAUpdateSQL) ;
```

**Remarks**

Use the TMyUpdateSQL 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

TObject  
[TCustomDAUpdateSQL](#)  
**TMyUpdateSQL**

© 1997-2012 Devart. All Rights Reserved.

[TMyUpdateSQL](#) 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.  |
| <a href="#">DeleteObject</a> (inherited from <a href="#">TCustomDAUpdateSQL</a> )  | Provides ability to perform advanced adjustment of the delete operations.   |
| <a href="#">DeleteSQL</a> (inherited from <a href="#">TCustomDAUpdateSQL</a> )     | Used when deleting a record.  |
| <a href="#">InsertObject</a> (inherited from <a href="#">TCustomDAUpdateSQL</a> )  | Provides ability to perform advanced adjustment of insert operations.   |
| <a href="#">InsertSQL</a> (inherited from <a href="#">TCustomDAUpdateSQL</a> )     | Used when inserting a record.   |
| <a href="#">LockObject</a> (inherited from <a href="#">TCustomDAUpdateSQL</a> )    | Provides ability to perform advanced adjustment of lock operations.   |
| <a href="#">LockSQL</a> (inherited from <a href="#">TCustomDAUpdateSQL</a> )       | Used to lock the current record.  |
| <a href="#">ModifyObject</a> (inherited from <a href="#">TCustomDAUpdateSQL</a> )  | Provides ability to perform advanced adjustment of modify operations.   |
| <a href="#">ModifySQL</a> (inherited from <a href="#">TCustomDAUpdateSQL</a> )     | Used when updating a record.  |
| <a href="#">RefreshObject</a> (inherited from <a href="#">TCustomDAUpdateSQL</a> ) | Provides ability to perform advanced adjustment of refresh operations.  |
| <a href="#">RefreshSQL</a> (inherited from <a href="#">TCustomDAUpdateSQL</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> (inherited from <a href="#">TCustomDAUpdateSQL</a> )           | Used to return a SQL statement for one of the <code>ModifySQL</code> , <code>InsertSQL</code> , or <code>DeleteSQL</code> 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.   |

© 1997-2012 Devart. All Rights Reserved.

## 17.15.2 Types

Types in the **MyAccess** unit.

### Types

| Name                                  | Description  |
|---------------------------------------|--|
| <a href="#">TMyUpdateExecuteEvent</a> | This type is used for the E:Devart.MyDac.TCustomMyDataSet.AfterUpdateExecute and E:Devart.MyDac.TCustomMyDataSet.BeforeUpdateExecute events. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.15.2.1 MyAccess.TMyUpdateExecuteEvent Procedure Reference

This type is used for the E:Devart.MyDac.TCustomMyDataSet.AfterUpdateExecute and E:Devart.MyDac.TCustomMyDataSet.BeforeUpdateExecute events.

### Unit

[MyAccess](#)

### Syntax

```
TMyUpdateExecuteEvent = procedure (Sender: TCustomMyDataSet;  
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.

© 1997-2012 Devart. All Rights Reserved.

### 17.15.3 Enumerations

Enumerations in the **MyAccess** unit.

#### Enumerations

| Name                              | Description  |
|-----------------------------------|--|
| <a href="#">TLockRecordType</a>   | Specifies the type of the record locking.  |
| <a href="#">TLockType</a>         | Specifies the type of the table locking.   |
| <a href="#">TMyIsolationLevel</a> | Specifies the extent to which all outside transactions interfere with subsequent transactions of current connection. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.15.3.1 MyAccess.TLockRecordType Enumeration

Specifies the type of the record locking.

##### Unit

[MyAccess](#)

##### Syntax

```
TLockRecordType = (lrImmediately, lrDelayed);
```

##### Values

| Value                | Meaning   |
|----------------------|---|
| <b>lrDelayed</b>     | Locking is performed just at the time of execution Post for this record.  |
| <b>lrImmediately</b> | Checks for locking directly on calling the Lock method. In this case if set Lock(lrImmediately) call to BeforeEdit event you can unable the same row in the table to be modified by several users at the same time. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.15.3.2 MyAccess.TLockType Enumeration

Specifies the type of the table locking.

##### Unit

[MyAccess](#)

##### Syntax

```
TLockType = (ltRead, ltReadLocal, ltWrite, ltWriteLowPriority);
```

##### Values

| Value                     | Meaning   |
|---------------------------|---|
| <b>ltRead</b>             | All connections including the current one can read only from the specified table.   |
| <b>ltReadLocal</b>        | Similar to ltRead. Moreover, allows non-conflicting INSERT statements to execute while the lock is held.  |
| <b>ltWrite</b>            | The current connection can read and write to the table. Other connections wait for the UnlockTable call.  |
| <b>ltWriteLowPriority</b> | You can use ltWriteLowPriority locks to allow other threads to obtain ltRead locks while the thread is waiting for the ltWrite lock. You should use only ltWriteLowPriority locks if you are sure that eventually there will be a time when no threads will have a ltRead lock. |

© 1997-2012 Devart. All Rights Reserved.

## 17.15.3.3 MyAccess.TMyIsolationLevel Enumeration

Specifies the extent to which all outside transactions interfere with subsequent transactions of current connection.

**Unit**

[MyAccess](#)

**Syntax**

```
TMyIsolationLevel = (ilReadCommitted, ilReadUnCommitted,  
    ilRepeatableRead, ilSerializable);
```

**Values**

| Value                    | Meaning   |
|--------------------------|---|
| <b>ilReadCommitted</b>   | Sets isolation level at which transaction cannot see changes made by outside transactions until they are committed. Only dirty reads (changes made by uncommitted transactions) are eliminated by this state of isolation level. The default value.   |
| <b>ilReadUnCommitted</b> | The most unrestricted level of transaction isolation. All types of data access interferences are possible. Mainly used for browsing database and to receive instant data with prospective changes.  |
| <b>ilRepeatableRead</b>  | Prevents concurrent transactions from modifying data in current uncommitted transaction. This level eliminates dirty reads as well as nonrepeatable reads (repeatable reads of the same data in one transaction before and after outside transactions may have started and committed).  |
| <b>ilSerializable</b>    | The most restricted level of transaction isolation. Database server isolates data involved in current transaction by putting additional processing on range locks. Used to put aside all undesired effects observed in concurrent accesses to the same set of data, but may lead to a greater latency at times of congested database environment. |

## 17.15.4 Routines

Routines in the **MyAccess** unit.

### Routines

| Name                          | Description   |
|-------------------------------|---|
| <a href="#">GetServerList</a> | Returns the list of the MySQL servers in LAN. MySQL server does not provide usual ways of such list getting, so it can be incomplete. |

---

© 1997-2012 Devart. All Rights Reserved.

#### 17.15.4.1 MyAccess.GetServerList Procedure

Returns the list of the MySQL servers in LAN. MySQL server does not provide usual ways of such list getting, so it can be incomplete.

### Unit

[MyAccess](#)

### Syntax

```
procedure GetServerList(List: TStrings);
```

#### Parameters

*List*

the list of the MySQL servers in LAN.

---

© 1997-2012 Devart. All Rights Reserved.



## 17.15.5 Constants

Constants in the **MyAccess** unit.

### Constants

| Name                         | Description   |
|------------------------------|---|
| <a href="#">MydacVersion</a> | Read this constant to get current version number for MyDAC. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.15.5.1 MyAccess.MydacVersion Constant

Read this constant to get current version number for MyDAC.

### Unit

[MyAccess](#)

### Syntax

```
MyDACVersion = '7.5.9';
```

© 1997-2012 Devart. All Rights Reserved.

## 17.16 MyBackup

This unit contains implementation of the TMyBackup component.

### Classes

| Name                      | Description  |
|---------------------------|--|
| <a href="#">TMyBackup</a> | A component that serves for backup copying specified tables on the server. |

### Types

| Name                             | Description   |
|----------------------------------|---|
| <a href="#">TMyTableMsgEvent</a> | This type is used for the <a href="#">TMyBackup.OnTableMsg</a> event. |

### Enumerations

| Name                                 | Description   |
|--------------------------------------|---|
| <a href="#">TMyBackupMode</a>        | Specifies the mode of TMyBackup work.   |
| <a href="#">TMyBackupPriority</a>    | Specifies priority of the <a href="#">TMyBackup.Restore</a> operation.  |
| <a href="#">TMyRestoreDuplicates</a> | Specifies the behaviour on detection records with repeated key fields on the execution of the <a href="#">TMyBackup.Restore</a> method. |

---

## 17.16.1 Classes

Classes in the **MyBackup** unit.

### Classes

| Name                      | Description  |
|---------------------------|--|
| <a href="#">TMyBackup</a> | A component that serves for backup copying specified tables on the server. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.16.1.1 MyBackup.TMyBackup Class

A component that serves for backup copying specified tables on the server.

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

### Unit

[MyBackup](#)

### Syntax

```
TMyBackup = class(TComponent);
```

### Remarks

Serves for backup copying specified tables on the server. Supports working in two modes defined by the [TMyBackup.Mode](#) property - bmText and bmBinary.

The list of tables is specified in [TMyBackup.TableNames](#).

Use the [TMyBackup.Path](#) property to specify the path to the server.

TMyBackup works on the server side that greatly affects on performance. However, it has some restrictions. Firstly, backup is performed only for the tables, database structure and user rights that are not stored. Secondly, files created on the server cannot be modified and deleted by MySQL tools.

**Note:** When using bmBinary mode, both servers must have the same format of the tables (must be the same version). About compatibility of formats storing data please see MySQL Reference Manual.

### Inheritance Hierarchy

TObject

**TMyBackup**

### See Also

- [TMyBackup.Backup](#)
- [TMyBackup.Restore](#)
- [TMyDump](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyBackup](#) class overview.

### Properties

| Name                           | Description   |
|--------------------------------|---|
| <a href="#">BackupPriority</a> | Specifies priority of the <a href="#">TMyBackup.Restore</a> operation.  |
| <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 the statement being executed.   |
| <a href="#">Duplicates</a>     | Used to specify the behaviour on detection records with repeated key fields on the execution of the <a href="#">TMyBackup.Restore</a> method. |

|                                    |   |
|------------------------------------|---|
| <a href="#">EnclosedBy</a>         | Used to specify a string (a character set) with which a field can be quoted.                      |
| <a href="#">EscapedBy</a>          | Used to specify a symbol that will be used as a special symbol pointer.                           |
| <a href="#">Fields</a>             | Holds the list of the fields that will be saved.  |
| <a href="#">FieldsTerminatedBy</a> | Used to specify field delimiter.  |
| <a href="#">IgnoreLines</a>        | Determines if the specified rows number at the beginning of the file will be ignored or not.      |
| <a href="#">LinesTerminatedBy</a>  | Used to specify row delimiter.  |
| <a href="#">Local</a>              | Used to specify that <a href="#">TMyBackup.Path</a> is a local path but not a path on the server. |
| <a href="#">Mode</a>               | Used to control modes of TMyBackup work.  |
| <a href="#">Path</a>               | Holds a path on the server to the folder where data files will be stored.                         |
| <a href="#">TableNames</a>         | Holds the list of tables which will be used in the script.  |

## Methods

| Name                    | Description                                  |
|-------------------------|--|
| <a href="#">Backup</a>  | Copies current tables by the specified path. |
| <a href="#">Restore</a> | Restores tables.                             |

## Events

| Name                       | Description   |
|----------------------------|---|
| <a href="#">OnTableMsg</a> | Occurs on executing of the <a href="#">TMyBackup.Restore</a> operation. |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyBackup** class.

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

## Published

| Name                           | Description   |
|--------------------------------|---|
| <a href="#">BackupPriority</a> | Specifies priority of the <a href="#">TMyBackup.Restore</a> operation.  |
| <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 the statement being executed.   |
| <a href="#">Duplicates</a>     | Used to specify the behaviour on detection records with repeated key fields on the execution of the <a href="#">TMyBackup.Restore</a> method. |
| <a href="#">EnclosedBy</a>     | Used to specify a string (a character set) with which a field can be quoted.  |
| <a href="#">EscapedBy</a>      | Used to specify a symbol that will be used as a special symbol pointer.   |
| <a href="#">Fields</a>         | Holds the list of the fields that will be saved.  |

[FieldsTerminatedBy](#)  
[IgnoreLines](#)

[LinesTerminatedBy](#)  
[Local](#)

[Mode](#)

[Path](#)

[TableNames](#)

Used to specify field delimiter.

Determines if the specified rows number at the beginning of the file will be ignored or not.

Used to specify row delimiter.

Used to specify that [TMyBackup.Path](#) is a local path but not a path on the server.

Used to control modes of TMyBackup work.

Holds a path on the server to the folder where data files will be stored.

Holds the list of tables which will be used in the script.

### See Also

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

---

© 1997-2012 Devart. All Rights Reserved.

Specifies priority of the [Restore](#) operation.

### Class

[TMyBackup](#)

### Syntax

```
property BackupPriority: TMyBackupPriority default bpDefault;
```

### Remarks

Specifies a priority on [Restore](#) operation.  
Used only if [Mode](#) is bmText.

### See Also

- [Mode](#)
- [Restore](#)

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify a connection object that will be used to connect to a data store.

### Class

[TMyBackup](#)

### Syntax

```
property Connection: TCustomMyConnection;
```

### 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 TCustomMyConnection objects.  
At run-time, set the Connection property to reference an existing TCustomMyConnection object.

### See Also

- [TCustomMyConnection](#)
-

© 1997-2012 Devart. All Rights Reserved.

Used to display the statement being executed.

### Class

[TMyBackup](#)

### Syntax

```
property Debug: boolean default False;
```

### Remarks

Set the Debug property to True to display the statement being executed.  
The default value is False.

**Note:** To enable debug form display you should explicitly include MyDacVcl (MyDacClx under Kylix) unit in your project.

### See Also

- [TCustomDASQL.Debug](#)
  - [TCustomDADataset.Debug](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to specify the behaviour on detection records with repeated key fields on the execution of the [Restore](#) method.

### Class

[TMyBackup](#)

### Syntax

```
property Duplicates: TMyRestoreDuplicates default bdError;
```

### Remarks

Use the Duplicates property to specify the behaviour on detection records with repeated key fields on the execution of the [Restore](#) method.  
Used only if [Mode](#) is bmText.

### See Also

- [Mode](#)
  - [Restore](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to specify a string (a character set) with which a field can be quoted.

### Class

[TMyBackup](#)

### Syntax

```
property EnclosedBy: string;
```

### Remarks

Use the EnclosedBy property to specify a string (a character set) with which a field can be quoted. By default it is an empty string.  
Used only if [Mode](#) is bmText.

## See Also

- [Backup](#)
- [Mode](#)
- [EscapedBy](#)
- [FieldsTerminatedBy](#)
- [LinesTerminatedBy](#)

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify a symbol that will be used as a special symbol pointer.

## Class

[TMyBackup](#)

## Syntax

**property** EscapedBy: **string**;

## Remarks

Use the EscapedBy property to specify a symbol that will be used as a special symbol pointer to define the entering of a special symbol. By default such pointer is '\\'.  
Used only if [Mode](#) is bmText.

## See Also

- [Backup](#)
- [Mode](#)
- [EnclosedBy](#)
- [FieldsTerminatedBy](#)
- [LinesTerminatedBy](#)

---

© 1997-2012 Devart. All Rights Reserved.

Holds the list of the fields that will be saved.

## Class

[TMyBackup](#)

## Syntax

**property** Fields: **string**;

## Remarks

The Fields property is used to contain the list of the fields that will be saved. If not specified, all fields will be stored.  
Use this property on execution [Backup](#) for a single table (see [TableNames](#)).  
Used only if [Mode](#) is bmText.

## See Also

- [Mode](#)
- [TableNames](#)
- [Backup](#)
- [Restore](#)

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify field delimiter.

### Class

[TMyBackup](#)

### Syntax

```
property FieldsTerminatedBy: string;
```

### Remarks

Use the FieldsTerminatedBy property to specify a string that will be used as field delimiter. By default field delimiter is '\t'.

Used only if [Mode](#) is bmText.

### See Also

- [Backup](#)
  - [Mode](#)
  - [EnclosedBy](#)
  - [EscapedBy](#)
  - [LinesTerminatedBy](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Determines if the specified rows number at the beginning of the file will be ignored or not.

### Class

[TMyBackup](#)

### Syntax

```
property IgnoreLines: integer default 0;
```

### Remarks

Use the IgnoreLines property to determine if the specified rows number at the beginning of the file will be ignored or not.

Use this property if you need:

- create non-typical formatted text file to be processed by the third-party applications
- fast load of big volume data to the database generated by the third-party application.

Used only if [Mode](#) is bmText.

### See Also

- [Mode](#)
  - [Restore](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to specify row delimiter.

### Class

[TMyBackup](#)

### Syntax

```
property LinesTerminatedBy: string;
```

### Remarks

Use the LinesTerminatedBy property to specify a string that will be used as row delimiter. By default row delimiter is '\n'.

Used only if [Mode](#) is bmText.



## See Also

- [Backup](#)
- [Mode](#)
- [EnclosedBy](#)
- [EscapedBy](#)
- [FieldsTerminatedBy](#)

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify that [Path](#) is a local path but not a path on the server.

## Class

[TMyBackup](#)

## Syntax

```
property Local: boolean default False;
```

## Remarks

Use the Local property to specify that [Path](#) is a local path but not a path on the server on the execution of the [Restore](#) method.

Used only if [Mode](#) is bmText.

## See Also

- [Mode](#)
- [Restore](#)

---

© 1997-2012 Devart. All Rights Reserved.

Used to control modes of TMyBackup work.

## Class

[TMyBackup](#)

## Syntax

```
property Mode: TMyBackupMode default bmBinary;
```

## Remarks

Use the Mode property to control modes of TMyBackup work.

**Note:** By security reasons MySQL does not allow to overwrite files that already exist. Also MySQL requires a path to the server to be created beforehand.

When using bmBinary mode, both servers must have the same format of the tables (must be the same version). About the compatibility of formats storing data please see MySQL Reference Manual.

## See Also

- [Backup](#)
- [Restore](#)

---

© 1997-2012 Devart. All Rights Reserved.

Holds a path on the server to the folder where data files will be stored.

## Class

[TMyBackup](#)

## Syntax

```
property Path: string;
```

### Remarks

Use the Path property to contain a path on the server to the folder where data files will be stored. Path must exist before executing the [Backup](#) and [Restore](#) operations.

### See Also

- [Backup](#)
- [Restore](#)

---

© 1997-2012 Devart. All Rights Reserved.

Holds the list of tables which will be used in the script.

### Class

[TMyBackup](#)

### Syntax

```
property TableNames: string;
```

### Remarks

Use the TableNames property to hold the list of tables which will be used in the script. Table names are separated by comma or semicolon. If it has an empty value, all the tables presented in the database will be processed.

### See Also

- [Backup](#)
- [Restore](#)
- M:Devart.Dac.TCustomDAConnection.GetTableNames(Borland.Vcl.TStrings,System.Boolean)

---

© 1997-2012 Devart. All Rights Reserved.

Methods of the **TMyBackup** class.

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

### Public

| Name                    | Description                                  |
|-------------------------|--|
| <a href="#">Backup</a>  | Copies current tables by the specified path. |
| <a href="#">Restore</a> | Restores tables.                             |

### See Also

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

---

© 1997-2012 Devart. All Rights Reserved.

Copies current tables by the specified path.

### Class

[TMyBackup](#)

### Syntax

```
procedure Backup;
```

### Remarks

Call the Backup method to copy current tables by the specified path (see [Path](#) property). A list of tables is specified in the [TableNames](#) property. If the list isn't specified, all tables will be copied. Use the [Mode](#) property to specify a copy mode. If Mode is bmBinary before backup copying of table files, tables are [TCustomMyDataSet.LockTable](#) on writing and [TMyServerControl.Flush](#) is called.

## See Also

- [Mode](#)
- [TableNames](#)
- [Path](#)

---

© 1997-2012 Devart. All Rights Reserved.

Restores tables.

## Class

[TMyBackup](#)

## Syntax

```
procedure Restore;
```

## Remarks

Call the Restore method to restore tables.

## See Also

- [OnTableMsg](#)

---

© 1997-2012 Devart. All Rights Reserved.

Events of the **TMyBackup** class.

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

## Published

| Name                       | Description   |
|----------------------------|---|
| <a href="#">OnTableMsg</a> | Occurs on executing of the <a href="#">TMyBackup.Restore</a> operation. |

## See Also

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

---

© 1997-2012 Devart. All Rights Reserved.

Occurs on executing of the [Restore](#) operation.

## Class

[TMyBackup](#)

## Syntax

```
property OnTableMsg: TMyTableMsgEvent;
```

## Remarks

The OnTableMsg event occurs on executing of the [Restore](#) operation.

## See Also

- [Restore](#)
- 

© 1997-2012 Devart. All Rights Reserved.

## 17.16.2 Types

Types in the **MyBackup** unit.

### Types

| Name                             | Description   |
|----------------------------------|---|
| <a href="#">TMyTableMsgEvent</a> | This type is used for the <a href="#">TMyBackup.OnTableMsg</a> event. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.16.2.1 MyBackup.TMyTableMsgEvent Procedure Reference

This type is used for the [TMyBackup.OnTableMsg](#) event.

### Unit

[MyBackup](#)

### Syntax

```
TMyTableMsgEvent = procedure (Sender: TObject; TableName: string;  
    MsgText: string) of object;
```

#### Parameters

##### *Sender*

An object that raised the event.

##### *TableName*

Holds the table name.

##### *MsgText*

Holds the message text.

© 1997-2012 Devart. All Rights Reserved.

## 17.16.3 Enumerations

Enumerations in the **MyBackup** unit.

### Enumerations

| Name                                 | Description   |
|--------------------------------------|---|
| <a href="#">TMyBackupMode</a>        | Specifies the mode of TMyBackup work.   |
| <a href="#">TMyBackupPriority</a>    | Specifies priority of the <a href="#">TMyBackup.Restore</a> operation.  |
| <a href="#">TMyRestoreDuplicates</a> | Specifies the behaviour on detection records with repeated key fields on the execution of the <a href="#">TMyBackup.Restore</a> method. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.16.3.1 MyBackup.TMyBackupMode Enumeration

Specifies the mode of TMyBackup work.

### Unit

[MyBackup](#)

### Syntax

```
TMyBackupMode = (bmBinary, bmText);
```

### Values

| Value           | Meaning   |
|-----------------|---|
| <b>bmBinary</b> | The fastest way of saving data. Files with the tables are copied physically to the specified folder (Path). This mode is not supported for several table types, for example, InnoDB.  |
| <b>bmText</b>   | Data from the tables is copied as a text file. To specify its format use <a href="#">TMyBackup.EnclosedBy</a> , <a href="#">TMyBackup.EscapedBy</a> , <a href="#">TMyBackup.FieldsTerminatedBy</a> , <a href="#">TMyBackup.LinesTerminatedBy</a> properties. This mode can be used for all table types. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.16.3.2 MyBackup.TMyBackupPriority Enumeration

Specifies priority of the [TMyBackup.Restore](#) operation.

### Unit

[MyBackup](#)

### Syntax

```
TMyBackupPriority = (bpDefault, bpLowPriority, bpConcurrent);
```

### Values

| Value                | Meaning   |
|----------------------|---|
| <b>bpConcurrent</b>  | Restore will be executed simultaneously with queries from other connections.                |
| <b>bpDefault</b>     | Other connections wait for finishing Restore.   |
| <b>bpLowPriority</b> | Execution of Restore will be suspended until other connections stop reading from the table. |

© 1997-2012 Devart. All Rights Reserved.

## 17.16.3.3 MyBackup.TMyRestoreDuplicates Enumeration

Specifies the behaviour on detection records with repeated key fields on the execution of the [TMyBackup.Restore](#) method.

**Unit**

[MyBackup](#)

**Syntax**

```
TMyRestoreDuplicates = (bdIgnore, bdReplace, bdError);
```

**Values**

| Value            | Meaning   |
|------------------|---|
| <b>bdError</b>   | Generate an error, ignore the rest of the file. |
| <b>bdIgnore</b>  | Ignore such records.                            |
| <b>bdReplace</b> | Replace old record with a new one.              |

## **17.17 MyBuilderClient**

This unit contains implementation of the TMyBuilder class.

### **Classes**

| <b>Name</b>                | <b>Description</b>                                     |
|----------------------------|--|
| <a href="#">TMyBuilder</a> | A component for managing SQL Builder for MySQL Add-in. |

---

© 1997-2012 Devart. All Rights Reserved.



## 17.17.1 Classes

Classes in the **MyBuilderClient** unit.

### Classes

| Name                       | Description  |
|----------------------------|--|
| <a href="#">TMyBuilder</a> | A component for managing SQL Builder for MySQL Add-in. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.17.1.1 MyBuilderClient.TMyBuilder Class

A component for managing SQL Builder for MySQL Add-in.

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

### Unit

[MyBuilderClient](#)

### Syntax

```
TMyBuilder = class (TComponent);
```

### Remarks

Serves to manage [SQL Builder for MySQL Add-in](#), free tool for visual building queries. The last version is available for download at <http://www.devart.com/mybuilder/mybuilderadd.exe>

Available only for Windows.

To use TMyBuilder in the applications on machines where SQL Builder for MySQL Add-in not installed, you must copy MyBuilder.dll file.

If the path to this file is not available for LoadLibrary, you need to register it by executing *regsvr32 MyBuilder.dll*

The full description of LoadLibrary you can see in MSDN, but the best is to place MyBuilder.dll in the same directory with your executable .exe file.

### Inheritance Hierarchy

TObject

**TMyBuilder**

### See Also

- [MyBuilder Add-In](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyBuilder](#) class overview.

### Properties

| Name                       | Description   |
|----------------------------|---|
| <a href="#">Connection</a> | Used to specify a connection object that will be used to connect to a server. |
| <a href="#">SQL</a>        | Used to provide a SQL statement to SQL Builder for MySQL Add-in.              |
| <a href="#">Version</a>    | Holds the version of SQL Builder for MySQL Add-in.                            |

### Methods

| Name                 | Description  |
|----------------------|--|
| <a href="#">Show</a> | Opens SQL Builder for MySQL Add-in in a modeless window. |

[ShowModal](#)

Opens SQL Builder for MySQL in a modal window.

---

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyBuilder** class.

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

**Public**

| Name                    | Description  |
|-------------------------|--|
| <a href="#">Version</a> | Holds the version of SQL Builder for MySQL Add-in. |

**Published**

| Name                       | Description   |
|----------------------------|---|
| <a href="#">Connection</a> | Used to specify a connection object that will be used to connect to a server. |
| <a href="#">SQL</a>        | Used to provide a SQL statement to SQL Builder for MySQL Add-in.              |

**See Also**

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

© 1997-2012 Devart. All Rights Reserved.

Used to specify a connection object that will be used to connect to a server.

**Class**

[TMyBuilder](#)

**Syntax**

**property** Connection: [TMyConnection](#);

**Remarks**

Use the Connection property to specify a connection object that will be used to connect to a server. Set at design-time by selecting from the list of provided TMyConnection objects. At run-time, set the Connection property to reference an existing TMyConnection object.

**See Also**

- [TCustomMyConnection](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to provide a SQL statement to SQL Builder for MySQL Add-in.

**Class**

[TMyBuilder](#)

**Syntax**

**property** SQL: TStrings;

**Remarks**

Use the SQL property to provide a SQL statement to SQL Builder for MySQL Add-in. At design time the SQL property can be edited by invoking a String List editor in the Object Inspector.

## See Also

- [Show](#)
- [ShowModal](#)

---

© 1997-2012 Devart. All Rights Reserved.

Holds the version of SQL Builder for MySQL Add-in.

## Class

[TMyBuilder](#)

## Syntax

```
property Version: string;
```

## Remarks

Holds the version of SQL Builder for MySQL Add-in.

## See Also

- [MyBuilder Add-In](#)

---

© 1997-2012 Devart. All Rights Reserved.

Methods of the **TMyBuilder** class.

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

## Public

| Name                      | Description  |
|---------------------------|--|
| <a href="#">Show</a>      | Opens SQL Builder for MySQL Add-in in a modeless window. |
| <a href="#">ShowModal</a> | Opens SQL Builder for MySQL in a modal window.           |

## See Also

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

---

© 1997-2012 Devart. All Rights Reserved.

Opens SQL Builder for MySQL Add-in in a modeless window.

## Class

[TMyBuilder](#)

## Syntax

```
procedure Show;
```

## Remarks

Call the Show method to open SQL Builder for MySQL Add-in in a modeless window. In case of absence of SQL Builder for MySQL a dialog window with an offer to download it will be displayed.

## See Also

- [SQL](#)
  - [ShowModal](#)
  - [MyBuilder Add-In](#)
-

© 1997-2012 Devart. All Rights Reserved.

Opens SQL Builder for MySQL in a modal window.

### Class

[TMyBuilder](#)

### Syntax

```
function ShowModal: boolean;
```

#### Return Value

True, if the SQL property has been changed.

### Remarks

Call the ShowModal method to open SQL Builder for MySQL in a modal window. In case of absence of SQL Builder for MySQL a dialog window with an offer to download it will be displayed.

### See Also

- [SQL](#)
  - [Show](#)
  - [MyBuilder Add-In](#)
- 

© 1997-2012 Devart. All Rights Reserved.

## **17.18 MyClasses**

This unit contains implementation of the EMyError class.

### **Classes**

| <b>Name</b>              | <b>Description</b>   |
|--------------------------|--|
| <a href="#">EMyError</a> | A base class that is raised when MySQL server returns error as a result. |

### **Enumerations**

| <b>Name</b>                 | <b>Description</b>   |
|-----------------------------|--|
| <a href="#">TMyProtocol</a> | Specifies which protocol to use when connecting to server. |

### **Variables**

| <b>Name</b>                         | <b>Description</b>  |
|-------------------------------------|---|
| <a href="#">_Strings65535ToMemo</a> | Control flow functions of MySQL (like IF, CASE) change data type of LONGMEMO and LONGBLOB fields. |

## 17.18.1 Classes

Classes in the **MyClasses** unit.

### Classes

| Name                     | Description  |
|--------------------------|--|
| <a href="#">EMyError</a> | A base class that is raised when MySQL server returns error as a result. |

---

© 1997-2012 Devart. All Rights Reserved.

#### 17.18.1.1 MyClasses.EMyError Class

A base class that is raised when MySQL server returns error as a result.

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

### Unit

[MyClasses](#)

### Syntax

```
EMyError = class (EDAError) ;
```

### Remarks

EMyError is raised when MySQL Server returns error as a result, for example, of an attempt to execute invalid SQL statement. Use EMyError in an exception-handling block.

### Inheritance Hierarchy

TObject

[EDAError](#)

**EMyError**

### See Also

- [EDAError](#)
- 

© 1997-2012 Devart. All Rights Reserved.

[EMyError](#) class overview.

### Properties

| Name   | Description   |
|--|---|
| <a href="#">Component</a> (inherited from <a href="#">EDAError</a> ) | Contains the component that caused the error.             |
| <a href="#">ErrorCode</a> (inherited from <a href="#">EDAError</a> ) | Determines the error code returned by the server.         |
| <a href="#">LineNumber</a>   | Contains the number of a query line that caused an error. |

---

© 1997-2012 Devart. All Rights Reserved.

Properties of the **EMyError** class.

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

### Public

| Name   | Description                                   |
|--|---|
| <a href="#">Component</a> (inherited from <a href="#">EDAError</a> ) | Contains the component that caused the error. |

[ErrorCode](#) (inherited from [EDAEError](#))

Determines the error code returned by the server.

[LineNumber](#)

Contains the number of a query line that caused an error.

### See Also

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

---

© 1997-2012 Devart. All Rights Reserved.

Contains the number of a query line that caused an error.

### Class

[EMyError](#)

### Syntax

```
property LineNumber: integer;
```

### Remarks

If an error, having [EDAEError.ErrorCode](#) = ER\_PARSE\_ERROR, occurred during query execution, LineNumber property contains the number of a query line that caused an error. MyDAC will retrieve this information from the error text automatically.

---

© 1997-2012 Devart. All Rights Reserved.

## 17.18.2 Enumerations

Enumerations in the **MyClasses** unit.

### Enumerations

| Name                        | Description  |
|-----------------------------|--|
| <a href="#">TMyProtocol</a> | Specifies which protocol to use when connecting to server. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.18.2.1 MyClasses.TMyProtocol Enumeration

Specifies which protocol to use when connecting to server.

### Unit

[MyClasses](#)

### Syntax

```
TMyProtocol = (mpDefault, mpTCP, mpSocket, mpPipe, mpMemory,
mpSSL, mpHttp);
```

### Values

| Value            | Meaning  |
|------------------|--|
| <b>mpDefault</b> | Similar to mpTCP, except the cases when you connect to a local server and the OS supports sockets (Unix) or named pipes (Windows), they are used instead of TCP/IP to connect to the server. |
| <b>mpHttp</b>    | Uses HTTP <a href="#">Network Tunneling</a> to connect to the server.  |
| <b>mpMemory</b>  | Uses SharedMem to connect to the server. Can be used with Direct set to False and libmysql.dll 4.1.  |
| <b>mpPipe</b>    | Uses NamedPipes to connect to the server.  |
| <b>mpSocket</b>  | Uses sockets to connect to the server. Can be used with Direct set to False and libmysql.dll 4.1.  |
| <b>mpSSL</b>     | Uses protected SSL connection with the server. To use SSL you need to set <a href="#">TMyConnection.SSLOptions</a> .   |
| <b>mpTCP</b>     | Uses TCP/IP to connect to the server.  |

© 1997-2012 Devart. All Rights Reserved.



### 17.18.3 Variables

Variables in the **MyClasses** unit.

#### Variables

| Name                                 | Description   |
|--------------------------------------|---|
| <a href="#">__Strings65535ToMemo</a> | Control flow functions of MySQL (like IF, CASE) change data type of LONGMEMO and LONGBLOB fields. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.18.3.1 MyClasses.\_\_Strings65535ToMemo Variable

Control flow functions of MySQL (like IF, CASE) change data type of LONGMEMO and LONGBLOB fields.

#### Unit

[MyClasses](#)

#### Syntax

```
__Strings65535ToMemo: boolean = True;
```

#### Remarks

Control flow functions of MySQL (like IF, CASE) change data type of LONGMEMO and LONGBLOB fields. It causes wrong description of these fields by MyDAC and truncating their data. To avoid these problems, MyDAC tries to restore the correct data type. This behaviour was introduced in MyDAC 5.10.0.9. To disable this behaviour, set the `__Strings65535ToMemo` variable to False.

© 1997-2012 Devart. All Rights Reserved.

## **17.19 MyConnectionPool**

This unit contains the TMyConnectionPoolManager class for managing connection pool.

### **Classes**

| <b>Name</b>                              | <b>Description</b>   |
|--|--|
| <a href="#">TMyConnectionPoolManager</a> | A class of methods that are used for managing MyDAC connection pool. |

---

© 1997-2012 Devart. All Rights Reserved.

## 17.19.1 Classes

Classes in the **MyConnectionPool** unit.

### Classes

| Name                                     | Description  |
|--|--|
| <a href="#">TMyConnectionPoolManager</a> | A class of methods that are used for managing MyDAC connection pool. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.19.1.1 MyConnectionPool.TMyConnectionPoolManager Class

A class of methods that are used for managing MyDAC connection pool.

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

### Unit

[MyConnectionPool](#)

### Syntax

```
TMyConnectionPoolManager = class (TCRConnectionPoolManager) ;
```

### Remarks

Use the TMyConnectionPoolManager methods to manage MyDAC connection pool.

### Inheritance Hierarchy

```
TObject  
  TCRConnectionPoolManager  
    TMyConnectionPoolManager
```

### See Also

- [Connection Pooling](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyConnectionPoolManager](#) class overview.

© 1997-2012 Devart. All Rights Reserved.

## **17.20 MyDacVcl**

This unit contains the visual constituent of MyDAC.

### **Classes**

| <b>Name</b>                      | <b>Description</b>   |
|----------------------------------|--|
| <a href="#">TMyConnectDialog</a> | A class that provides a dialog box for user to supply his login information. |

---

## 17.20.1 Classes

Classes in the **MyDacVcl** unit.

### Classes

| Name                             | Description  |
|----------------------------------|--|
| <a href="#">TMyConnectDialog</a> | A class that provides a dialog box for user to supply his login information. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.20.1.1 MyDacVcl.TMyConnectDialog 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 [TMyConnectDialog](#) members.

### Unit

[MyDacVcl](#)

### Syntax

```
TMyConnectDialog = class (TCustomConnectDialog) ;
```

### Remarks

The TMyConnectDialog component is a direct descendant of TCustomConnectDialog class. Use TMyConnectDialog 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

```
TObject  
  TCustomConnectDialog  
    TMyConnectDialog
```

### See Also

- [TCustomDAConnection.ConnectDialog](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyConnectDialog](#) 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>   | Holds the TMyConnection component which uses TMyConnectDialog 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="#">PortLabel</a>  | Used to specify a prompt for port edit.   |
| <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.  |
| <a href="#">ShowDatabase</a>   | Used to display a field for entering database at connect dialog.  |
| <a href="#">ShowPort</a>   | Used to display a field for entering port at connect dialog.  |
| <a href="#">StoreLogInfo</a> (inherited from <a href="#">TCustomConnectDialog</a> )  | Used to specify whether the login information should be kept in system registry after a connection was established. |
| <a href="#">UsernameLabel</a> (inherited from <a href="#">TCustomConnectDialog</a> ) | 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.  |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyConnectDialog** class.

For a complete list of the **TMyConnectDialog** class members, see the [TMyConnectDialog 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>   | Holds the TMyConnection component which uses TMyConnectDialog 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.   |

|  |   |
|--|---|
| <a href="#">PasswordLabel</a> (inherited from <a href="#">TCustomConnectDialog</a> ) | Used to specify a prompt for password edit.   |
| <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.  |
| <a href="#">StoreLogInfo</a> (inherited from <a href="#">TCustomConnectDialog</a> )  | Used to specify whether the login information should be kept in system registry after a connection was established. |
| <a href="#">UsernameLabel</a> (inherited from <a href="#">TCustomConnectDialog</a> ) | 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="#">PortLabel</a>     | Used to specify a prompt for port edit.                          |
| <a href="#">ShowDatabase</a>  | Used to display a field for entering database at connect dialog. |
| <a href="#">ShowPort</a>      | Used to display a field for entering port at connect dialog.     |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Holds the TMyConnection component which uses TMyConnectDialog object.

## Class

[TMyConnectDialog](#)

## Syntax

**property** Connection: [TCustomMyConnection](#);

## Remarks

Use the Connection property to learn which TMyConnection component uses TMyConnectDialog object. This property is read-only.

## See Also

- [TCustomDAConnection.ConnectDialog](#)

© 1997-2012 Devart. All Rights Reserved.

Used to specify a prompt for database edit.

## Class

[TMyConnectDialog](#)

## Syntax

**property** DatabaseLabel: **string**;

### Remarks

Use the DatabaseLabel property to specify a prompt for database edit.

### See Also

- [ShowDatabase](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to specify a prompt for port edit.

### Class

[TMyConnectDialog](#)

### Syntax

```
property PortLabel: string;
```

### Remarks

Use the PortLabel property to specify a prompt for port edit.

### See Also

- [ShowPort](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to display a field for entering database at connect dialog.

### Class

[TMyConnectDialog](#)

### Syntax

```
property ShowDatabase: boolean default True;
```

### Remarks

Use the ShowDatabase property to display a field for entering database at connect dialog.  
The default value is True.

### See Also

- [DatabaseLabel](#)
  - [ShowPort](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to display a field for entering port at connect dialog.

### Class

[TMyConnectDialog](#)

### Syntax

```
property ShowPort: boolean default True;
```

### Remarks

Use the ShowPort property to display a field for entering port at connect dialog.  
The default value is True.



**See Also**

- [PortLabel](#)
- [ShowDatabase](#)

---

© 1997-2012 Devart. All Rights Reserved.

## 17.21 MyDump

This unit contains implementation of the TMyDump component.

### Classes

| Name                           | Description   |
|--------------------------------|---|
| <a href="#">TMyDump</a>        | A component for storing database or its parts as a script and also for restoring database from the received script. |
| <a href="#">TMyDumpOptions</a> | This class allows setting up the behaviour of the TMyDump class.  |

### Types

| Name                           | Description   |
|--------------------------------|---|
| <a href="#">TMyDumpObjects</a> | Represents the set of <a href="#">TMyDumpObject</a> . |

### Enumerations

| Name                          | Description                       |
|-------------------------------|-----------------------------------|
| <a href="#">TMyDumpObject</a> | Specifies the object enumeration. |

---

## 17.21.1 Classes

Classes in the **MyDump** unit.

### Classes

| Name                           | Description   |
|--------------------------------|---|
| <a href="#">TMyDump</a>        | A component for storing database or its parts as a script and also for restoring database from the received script. |
| <a href="#">TMyDumpOptions</a> | This class allows setting up the behaviour of the TMyDump class.  |

© 1997-2012 Devart. All Rights Reserved.

#### 17.21.1.1 MyDump.TMyDump Class

A component for storing database or its parts as a script and also for restoring database from the received script.

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

### Unit

[MyDump](#)

### Syntax

```
TMyDump = class (TDADump) ;
```

### Remarks

Serves to store a database or its parts as a script and also to restore database from received script.

TMyDump behaviour is similar to mysqldump program.

Use [TMyDump.Objects](#) and [TDADump.TableNames](#) properties to specify the list of objects to be stored. By default, only tables structure and data are stored.

To generate a script call [TDADump.Backup](#) or [TDADump.BackupQuery](#) method. Resulted script can be viewed in [TDADump.SQL](#).

TMyDump works on the client side. It causes large network loading. To backup database on the server side use [TMyBackup](#). Unlike TMyBackup, TMyDump component allows to store not only tables, but database structure including users' rights.

### Inheritance Hierarchy

TObject

[TDADump](#)

**TMyDump**

### See Also

- [TDADump.Backup](#)
- [TDADump.Restore](#)
- [TMyBackup](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyDump](#) 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> (inherited from <a href="#">TDADump</a> ) | Used to display executing statement, all its parameters' values, and the type of parameters. |

[Objects](#)[Options](#)[SQL](#) (inherited from [TDADump](#))[StoredProcNames](#)[TableNames](#) (inherited from [TDADump](#))[TriggerNames](#)

Used to set the enumeration of object types that will be described on calling [TDADump.Backup](#).

Specifies the behaviour of the TMyDump component.

Used to set or get the dump script.

Holds the list of procedures which will be used in the script.

Used to set the names of the tables to dump.

Holds the list of triggers which will be used in the script.

**Methods****Name**[Backup](#) (inherited from [TDADump](#))[BackupQuery](#) (inherited from [TDADump](#))[BackupToFile](#) (inherited from [TDADump](#))[BackupToStream](#) (inherited from [TDADump](#))[Restore](#) (inherited from [TDADump](#))[RestoreFromFile](#) (inherited from [TDADump](#))[RestoreFromStream](#) (inherited from [TDADump](#))**Description**

Dumps database objects to the [TDADump.SQL](#) property.

Dumps the results of a particular query.

Dumps database objects to the specified file.

Dumps database objects to the stream.

Executes a script contained in the SQL property.

Executes a script from a file.

Executes a script received from the stream.

**Events****Name**[OnBackupProgress](#) (inherited from [TDADump](#))[OnError](#) (inherited from [TDADump](#))[OnRestoreProgress](#) (inherited from [TDADump](#))**Description**

Occurs to indicate the [TDADump.Backup](#), M:Devart.Dac.TDADump.BackupToFile(System.String) or M:Devart.Dac.TDADump.BackupToStream(Borland.Vcl.TStream) method execution progress.

Occurs when MySQL raises some error on [TDADump.Restore](#).

Occurs to indicate the [TDADump.Restore](#), [TDADump.RestoreFromFile](#), or [TDADump.RestoreFromStream](#) method execution progress.

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyDump** class.

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

**Public****Name**[Backup](#) (inherited from [TDADump](#))[BackupQuery](#) (inherited from [TDADump](#))[BackupToFile](#) (inherited from [TDADump](#))**Description**

Dumps database objects to the [TDADump.SQL](#) property.

Dumps the results of a particular query.

Dumps database objects to the specified file.

[BackupToStream](#) (inherited from [TDADump](#))

Dumps database objects to the stream.

[Restore](#) (inherited from [TDADump](#))

Executes a script contained in the SQL property.

[RestoreFromFile](#) (inherited from [TDADump](#))

Executes a script from a file.

[RestoreFromStream](#) (inherited from [TDADump](#))

Executes a script received from the stream.

## Published

### Name

### Description

[Connection](#)

Used to specify a connection object that will be used to connect to a data store.

[Debug](#) (inherited from [TDADump](#))

Used to display executing statement, all its parameters' values, and the type of parameters.

[Objects](#)

Used to set the enumeration of object types that will be described on calling [TDADump.Backup](#).

[OnBackupProgress](#) (inherited from [TDADump](#))

Occurs to indicate the [TDADump.Backup](#), M:Devart.Dac.TDADump.BackupToFile(System.String) or M:Devart.Dac.TDADump.BackupToStream(Borland.Vcl.TStream) method execution progress.

[OnError](#) (inherited from [TDADump](#))

Occurs when MySQL raises some error on [TDADump.Restore](#).

[OnRestoreProgress](#) (inherited from [TDADump](#))

Occurs to indicate the [TDADump.Restore](#), [TDADump.RestoreFromFile](#), or [TDADump.RestoreFromStream](#) method execution progress.

[Options](#)

Specifies the behaviour of the TMyDump component.

[SQL](#) (inherited from [TDADump](#))

Used to set or get the dump script.

[StoredProcNames](#)

Holds the list of procedures which will be used in the script.

[TableNames](#) (inherited from [TDADump](#))

Used to set the names of the tables to dump.

[TriggerNames](#)

Holds the list of triggers which will be used in the script.

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Used to specify a connection object that will be used to connect to a data store.

## Class

[TMyDump](#)

## Syntax

**property** Connection: [TCustomMyConnection](#);

## 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 TCustomMyConnection objects. At runtime, set Connection property to reference an existing TCustomMyConnection object.

## See Also

- [TCustomMyConnection](#)

© 1997-2012 Devart. All Rights Reserved.

Used to set the enumeration of object types that will be described on calling [TDADump.Backup](#).

## Class

[TMyDump](#)

## Syntax

```
property Objects: TMyDumpObjects default [doTables, doViews, doData];
```

## Remarks

Use the Object property to set the enumeration of object types that will be described on calling [TDADump.Backup](#).

## See Also

- [TDADump.Backup](#)
- M:Devart.Dac.TDADump.BackupToFile(System.String,System.Boolean)
- M:Devart.Dac.TDADump.BackupToStream(Borland.Vcl.TStream,System.Boolean)
- [TDADump.TableNames](#)
- [StoredProcNames](#)

© 1997-2012 Devart. All Rights Reserved.

Specifies the behaviour of the TMyDump component.

## Class

[TMyDump](#)

## Syntax

```
property Options: TMyDumpOptions;
```

## Remarks

Set the properties of Options to specify the behaviour of a TMyDump component. Descriptions of all options are in the table below.

| Option Name                       | Description   |
|-----------------------------------|---|
| <a href="#">AddLock</a>           | Used to execute LOCK TABLE before data insertion.   |
| <a href="#">CommitBatchSize</a>   | Used to add COMMIT statement to script after inserting every CommitBatchSize strings when dumping table data. |
| <a href="#">DisableKeys</a>       | Allows disabling keys check when inserting records.   |
| <a href="#">HexBlob</a>           | Used to present BLOB values in hexadecimal notation.  |
| <a href="#">UseDelayedInsert</a>  | Set to use INSERT DELAYED.  |
| <a href="#">UseExtendedSyntax</a> | Set to use extended syntax of INSERT on data insertion.   |

## See Also

- 

## [Objects](#)

---

©

1997-2012 Devart. All Rights Reserved.

Holds the list of procedures which will be used in the script.

### Class

[TMyDump](#)

### Syntax

```
property StoredProcNames: string;
```

### Remarks

Use the StoredProcNames property to hold the list of procedures which will be used in the script. Makes sense only on execution [TDADump.Backup](#), M:Devart.Dac.TDADump.BackupToFile(System.String), M:Devart.Dac.TDADump.BackupToStream(Borland.Vcl.TStream) and doStoredProcs set at the [Objects](#). Names are separated by comma or semicolon. If it has an empty value, all procs presented in the database will be processed.

### See Also

- [TDADump.Backup](#)
- M:Devart.Dac.TDADump.BackupToFile(System.String,System.Boolean)
- M:Devart.Dac.TDADump.BackupToStream(Borland.Vcl.TStream,System.Boolean)
- [Objects](#)
- [TCustomDAConnection.GetStoredProcNames](#)

---

© 1997-2012 Devart. All Rights Reserved.

Holds the list of triggers which will be used in the script.

### Class

[TMyDump](#)

### Syntax

```
property TriggerNames: string;
```

### Remarks

Use the TriggerNames property to hold the list of triggers which will be used in the script. Makes sense only on execution of TMyDump.Backup, TMyDump.BackupToFile, TMyDump.BackupToStream, and doTriggers set at the Objects. Names are separated by comma or semicolon. If it has an empty value and the TMyDump.TableNames property is empty, then all triggers presented in the database will be processed. If it has an empty value and the TMyDump.TableNames property is not empty, then all triggers for the specified tables will be processed.

---

© 1997-2012 Devart. All Rights Reserved.

#### 17.21.1.2 MyDump.TMyDumpOptions Class

This class allows setting up the behaviour of the TMyDump class.

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

### Unit

[MyDump](#)

### Syntax

```
TMyDumpOptions = class(TDADumpOptions);
```

## Inheritance Hierarchy

```
TObject
  TDADumpOptions
    TMyDumpOptions
```

© 1997-2012 Devart. All Rights Reserved.

[TMyDumpOptions](#) class overview.

## Properties

| Name  | Description   |
|---|---|
| <a href="#">AddDrop</a> (inherited from <a href="#">TDADumpOptions</a> )        | Used to add drop statements to a script before creating statements.   |
| <a href="#">AddLock</a>   | Used to execute LOCK TABLE before data insertion.   |
| <a href="#">CommitBatchSize</a>   | Used to add COMMIT statement to script after inserting every CommitBatchSize strings when dumping table data. |
| <a href="#">DisableKeys</a>   | Allows disabling keys check when inserting records.   |
| <a href="#">GenerateHeader</a> (inherited from <a href="#">TDADumpOptions</a> ) | Used to add a comment header to a script.   |
| <a href="#">HexBlob</a>   | Used to present BLOB values in hexadecimal notation.  |
| <a href="#">QuoteNames</a> (inherited from <a href="#">TDADumpOptions</a> )     | Used for TDADump to quote all database object names in generated SQL statements.                              |
| <a href="#">UseDelayedIns</a>   | Set to use INSERT DELAYED.  |
| <a href="#">UseExtSyntax</a>  | Set to use extended syntax of INSERT on data insertion.   |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyDumpOptions** class.

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

## Published

| Name  | Description   |
|---|---|
| <a href="#">AddDrop</a> (inherited from <a href="#">TDADumpOptions</a> )        | Used to add drop statements to a script before creating statements.   |
| <a href="#">AddLock</a>   | Used to execute LOCK TABLE before data insertion.   |
| <a href="#">CommitBatchSize</a>   | Used to add COMMIT statement to script after inserting every CommitBatchSize strings when dumping table data. |
| <a href="#">DisableKeys</a>   | Allows disabling keys check when inserting records.   |
| <a href="#">GenerateHeader</a> (inherited from <a href="#">TDADumpOptions</a> ) | Used to add a comment header to a script.   |
| <a href="#">HexBlob</a>   | Used to present BLOB values in hexadecimal notation.  |
| <a href="#">QuoteNames</a> (inherited from <a href="#">TDADumpOptions</a> )     | Used for TDADump to quote all database object names in generated SQL statements.                              |
| <a href="#">UseDelayedIns</a>   | Set to use INSERT DELAYED.  |



[UseExtSyntax](#)

Set to use extended syntax of  
INSERT on data insertion.

### See Also

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

---

© 1997-2012 Devart. All Rights Reserved.

Used to execute LOCK TABLE before data insertion.

### Class

[TMyDumpOptions](#)

### Syntax

**property** AddLock: boolean **default** True;

### Remarks

Use the AddLock property to execute LOCK TABLE before data insertion. Used only with doData in [TMyDump.Objects](#).

---

© 1997-2012 Devart. All Rights Reserved.

Used to add COMMIT statement to script after inserting every CommitBatchSize strings when dumping table data.

### Class

[TMyDumpOptions](#)

### Syntax

**property** CommitBatchSize: integer **default** 0;

### Remarks

Use the CommitBatchSize property to add COMMIT statement to script after inserting every CommitBatchSize strings when dumping table data. Use this property to boost the productivity when recovering large amounts of data.

---

© 1997-2012 Devart. All Rights Reserved.

Allows disabling keys check when inserting records.

### Class

[TMyDumpOptions](#)

### Syntax

**property** DisableKeys: boolean **default** False;

### Remarks

Add `/*!40000 ALTER TABLE ... DISABLE KEYS */` before inserting data. Used only with doData in [TMyDump.Objects](#).

---

© 1997-2012 Devart. All Rights Reserved.

Used to present BLOB values in hexadecimal notation.

### Class

[TMyDumpOptions](#)

### Syntax

```
property HexBlob: boolean default False;
```

### Remarks

If the HexBlob property is True, the BLOB values are presented in hexadecimal notation.

---

© 1997-2012 Devart. All Rights Reserved.

Set to use INSERT DELAYED.

### Class

[TMyDumpOptions](#)

### Syntax

```
property UseDelayedIns: boolean default False;
```

### Remarks

Set the UseDelayedIns property to use INSERT DELAYED. Used only with doData in [TMyDump.Objects](#).

---

© 1997-2012 Devart. All Rights Reserved.

Set to use extended syntax of INSERT on data insertion.

### Class

[TMyDumpOptions](#)

### Syntax

```
property UseExtSyntax: boolean default True;
```

### Remarks

Set the UseExtSyntax property to use extended syntax of INSERT on data insertion. Used only with doData in [TMyDump.Objects](#).

---

© 1997-2012 Devart. All Rights Reserved.

## 17.21.2 Types

Types in the **MyDump** unit.

### Types

| Name                           | Description   |
|--------------------------------|---|
| <a href="#">TMyDumpObjects</a> | Represents the set of <a href="#">TMyDumpObject</a> . |

© 1997-2012 Devart. All Rights Reserved.

#### 17.21.2.1 MyDump.TMyDumpObjects Set

Represents the set of [TMyDumpObject](#).

### Unit

[MyDump](#)

### Syntax

```
TMyDumpObjects = set of TMyDumpObject;
```

© 1997-2012 Devart. All Rights Reserved.

### 17.21.3 Enumerations

Enumerations in the **MyDump** unit.

#### Enumerations

| Name                          | Description                       |
|-------------------------------|-----------------------------------|
| <a href="#">TMyDumpObject</a> | Specifies the object enumeration. |

© 1997-2012 Devart. All Rights Reserved.

17.21.3.1 MyDump.TMyDumpObject Enumeration  
Specifies the object enumeration.

#### Unit

[MyDump](#)

#### Syntax

```
TMyDumpObject = (doDatabase, doUsers, doStoredProcs, doTables,
doData, doViews, doTriggers);
```

#### Values

| Value                | Meaning  |
|----------------------|--|
| <b>doData</b>        | Stores data from the tables. The list of the tables is specified in <a href="#">TDADump.TableNames</a> . If TableNames is not set, all tables are used.  |
| <b>doDatabase</b>    | Adds to the received script "CREATE DATABASE..." command.  |
| <b>doStoredProcs</b> | Stores stored procedures structure in the script. The list of procedures is specified in <a href="#">TMyDump.StoredProcNames</a> . If StoredProcNames is not set, all procedures are used.   |
| <b>doTables</b>      | Stores tables structure in the script. The list of tables is specified in <a href="#">TDADump.TableNames</a> . If TableNames is not set, all tables are used.  |
| <b>doTriggers</b>    | Stores queries for creating triggers in the script. The list of triggers is specified in <a href="#">TMyDump.TriggerNames</a> . If TriggerNames is not set and the TMyDump.TableNames property is empty, then all triggers of the database will be processed. If TriggerNames is not set and the TMyDump.TableNames property is not empty, then all triggers for the specified tables will be processed. |
| <b>doUsers</b>       | Stores user privileges in the script.  |
| <b>doViews</b>       | Stores queries for creating Views in a script. The Views list is taken from TableNames.  |

© 1997-2012 Devart. All Rights Reserved.

## **17.22 MyEmbConnection**

This unit contains implementation of the TMyEmbConnection component.

### **Classes**

| <b>Name</b>                             | <b>Description</b>  |
|---|---|
| <a href="#"><u>TMyEmbConnection</u></a> | A component for establishing connection to Embedded MySQL server. |

## 17.22.1 Classes

Classes in the **MyEmbConnection** unit.

### Classes

| Name                             | Description   |
|----------------------------------|---|
| <a href="#">TMyEmbConnection</a> | A component for establishing connection to Embedded MySQL server. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.22.1.1 MyEmbConnection.TMyEmbConnection Class

A component for establishing connection to Embedded MySQL server.

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

### Unit

[MyEmbConnection](#)

### Syntax

```
TMyEmbConnection = class (TCustomMyConnection) ;
```

### Remarks

TMyEmbConnection component is used to establish a connection to MySQL Embedded server and provide its enhanced support.

Using TMyEmbConnection allows refusing from using configuration file (my.ini) and also lets you handle MySQL server messages written into log files.

### Inheritance Hierarchy

TObject

[TCustomDAConnection](#)

[TCustomMyConnection](#)

**TMyEmbConnection**

### See Also

- [TCustomMyConnection](#)
- [TMyConnection](#)
- [Embedded Server](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyEmbConnection](#) class overview.

### Properties

| Name  | Description  |
|---|--|
| <a href="#">BaseDir</a>   | Used to set the base path for MySQL Embedded server.                     |
| <a href="#">ClientVersion</a> (inherited from <a href="#">TCustomMyConnection</a> )     | Contains the version of the MySQL Client library.                        |
| <a href="#">ConnectDialog</a> (inherited from <a href="#">TCustomDAConnection</a> )     | Allows to link a <a href="#">TCustomConnectDialog</a> component.         |
| <a href="#">ConnectionTimeout</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Used to specify the amount of time to attempt to establish a connection. |
| <a href="#">ConvertEOL</a> (inherited from <a href="#">TCustomDAConnection</a> )        | Allows customizing line breaks in string fields and parameters.          |

|  |  |
|--|--|
| <a href="#">Database</a> (inherited from <a href="#">TCustomMyConnection</a> )       | Used to specify a database name that is a default source of data for SQL queries once a connection is established.     |
| <a href="#">DataDir</a>  | Used to set the path to the data directory.  |
| <a href="#">InTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )  | Indicates whether the transaction is active.   |
| <a href="#">IsolationLevel</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Used to specify extent to which all outside transactions interfere with subsequent transactions of current connection. |
| <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> (inherited from <a href="#">TCustomMyConnection</a> )        | Specifies the behaviour of the TMyConnectionOptions object.  |
| <a href="#">Params</a>   | Used to specify the list of command line parameters for Embedded server.   |
| <a href="#">Password</a> (inherited from <a href="#">TCustomDAConnection</a> )       | Serves to supply a password for login.   |
| <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> (inherited from <a href="#">TCustomDAConnection</a> )         | Serves to supply the server name for login.  |
| <a href="#">ServerVersion</a> (inherited from <a href="#">TCustomMyConnection</a> )  | Holds the version of MySQL server.   |
| <a href="#">ThreadId</a> (inherited from <a href="#">TCustomMyConnection</a> )       | Used to return the thread ID of the current connection.  |
| <a href="#">Username</a> (inherited from <a href="#">TCustomDAConnection</a> )       | Used to supply a user name for login.  |

## Methods

| Name  | Description   |
|---|---|
| <a href="#">ApplyUpdates</a> (inherited from <a href="#">TCustomDAConnection</a> )  | Overloaded. Applies changes in datasets.  |
| <a href="#">AssignConnect</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Shares database connection between the TCustomMyConnection components.                                |
| <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="#">CreateDataSet</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Returns a new instance of TCustomMyDataSet class and associates it with this connection object.       |
| <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="#">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="#">TCustomMyConnection</a> )       | Executes any SQL statement outside <a href="#">TMyQuery</a> or <a href="#">TMyCommand</a> components. |

|   |   |
|---|---|
| <a href="#">ExecSQLEx</a> (inherited from <a href="#">TCustomDAConnection</a> )           | Executes any SQL statement outside the TQuery or TSQL components.                                       |
| <a href="#">GetCharsetNames</a> (inherited from <a href="#">TCustomMyConnection</a> )     | Populates a string list with the names of available charsets.   |
| <a href="#">GetDatabaseNames</a> (inherited from <a href="#">TCustomDAConnection</a> )    | Returns a database list from the server.  |
| <a href="#">GetExecuteInfo</a> (inherited from <a href="#">TCustomMyConnection</a> )      | Returns the result of the last query execution.   |
| <a href="#">GetStoredProcNames</a> (inherited from <a href="#">TCustomDAConnection</a> )  | Returns a list of stored procedures from the server.  |
| <a href="#">GetTriggerNames</a> (inherited from <a href="#">TCustomMyConnection</a> )     | Returns a list of triggers from the server.   |
| <a href="#">MonitorMessage</a> (inherited from <a href="#">TCustomDAConnection</a> )      | Sends a specified message through the <a href="#">TCustomDASQLMonitor</a> component.                    |
| <a href="#">Ping</a> (inherited from <a href="#">TCustomMyConnection</a> )                | Allows to avoid automatic disconnection of the client by the server.                                    |
| <a href="#">ReleaseSavepoint</a> (inherited from <a href="#">TCustomMyConnection</a> )    | Releases the specified savepoint without affecting any work that has been performed after its creation. |
| <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="#">RollbackToSavepoint</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Cancels all updates for the current transaction.  |
| <a href="#">Savepoint</a> (inherited from <a href="#">TCustomMyConnection</a> )           | Defines a point in the transaction to which you can roll back later.                                    |
| <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.                   |
| <a href="#">OnError</a> (inherited from <a href="#">TCustomDAConnection</a> )          | This event occurs when an error has arisen in the connection. |
| <a href="#">OnLog</a>  | Occurs when MySQL server writes down to General Query Log.    |
| <a href="#">OnLogError</a>   | Occurs when MySQL server writes down to Error Log.            |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyEmbConnection** class.

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

## Public

| Name  | Description  |
|---|--|
| <a href="#">ApplyUpdates</a> (inherited from <a href="#">TCustomDAConnection</a> )  | Overloaded. Applies changes in datasets.                               |
| <a href="#">AssignConnect</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Shares database connection between the TCustomMyConnection components. |
| <a href="#">ClientVersion</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Contains the version of the MySQL Client library.                      |



|  |  |
|--|--|
| <a href="#"><u>Commit</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )             | Commits current transaction.   |
| <a href="#"><u>Connect</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )            | Establishes a connection to the server.  |
| <a href="#"><u>ConnectDialog</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )      | Allows to link a <a href="#"><u>TCustomConnectDialog</u></a> component.  |
| <a href="#"><u>ConnectionTimeout</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )  | Used to specify the amount of time to attempt to establish a connection.   |
| <a href="#"><u>ConvertEOL</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )         | Allows customi ing line breaks in string fields and parameters.  |
| <a href="#"><u>CreateDataSet</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )      | Returns a new instance of TCustomMyDataSet class and associates it with this connection object.                        |
| <a href="#"><u>CreateSQL</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )          | Creates a component for queries execution.   |
| <a href="#"><u>Database</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )           | Used to specify a database name that is a default source of data for SQL queries once a connection is established.     |
| <a href="#"><u>Disconnect</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )         | Performs disconnect.   |
| <a href="#"><u>ExecProc</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )           | Allows to execute stored procedure or function providing its name and parameters.                                      |
| <a href="#"><u>ExecProcEx</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )         | Allows to execute a stored procedure or function.  |
| <a href="#"><u>ExecSQL</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )            | Executes any SQL statement outside <a href="#"><u>TMyQuery</u></a> or <a href="#"><u>TMyCommand</u></a> components.    |
| <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>GetCharsetNames</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )    | Populates a string list with the names of available charsets.  |
| <a href="#"><u>GetDatabaseNames</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )   | Returns a database list from the server.   |
| <a href="#"><u>GetExecuteInfo</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )     | Returns the result of the last query execution.  |
| <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>GetTriggerNames</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )    | Returns a list of triggers from the server.  |
| <a href="#"><u>InTransaction</u></a> (inherited from <a href="#"><u>TCustomDAConnection</u></a> )      | Indicates whether the transaction is active.   |
| <a href="#"><u>IsolationLevel</u></a> (inherited from <a href="#"><u>TCustomMyConnection</u></a> )     | Used to specify extent to which all outside transactions interfere with subsequent transactions of current connection. |
| <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>TCustomMyConnection</u></a> )            | Specifies the behaviour of the TMyConnectionOptions object.  |

|   |   |
|---|---|
| <a href="#">Password</a> (inherited from <a href="#">TCustomDAConnection</a> )            | Serves to supply a password for login.  |
| <a href="#">Ping</a> (inherited from <a href="#">TCustomMyConnection</a> )                | Allows to avoid automatic disconnection of the client by the server.                                    |
| <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="#">ReleaseSavepoint</a> (inherited from <a href="#">TCustomMyConnection</a> )    | Releases the specified savepoint without affecting any work that has been performed after its creation. |
| <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="#">RollbackToSavepoint</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Cancels all updates for the current transaction.  |
| <a href="#">Savepoint</a> (inherited from <a href="#">TCustomMyConnection</a> )           | Defines a point in the transaction to which you can roll back later.                                    |
| <a href="#">Server</a> (inherited from <a href="#">TCustomDAConnection</a> )              | Serves to supply the server name for login.   |
| <a href="#">ServerVersion</a> (inherited from <a href="#">TCustomMyConnection</a> )       | Holds the version of MySQL server.  |
| <a href="#">StartTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )    | Begins a new user transaction.  |
| <a href="#">ThreadId</a> (inherited from <a href="#">TCustomMyConnection</a> )            | Used to return the thread ID of the current connection.   |
| <a href="#">Username</a> (inherited from <a href="#">TCustomDAConnection</a> )            | Used to supply a user name for login.   |

## Published

| Name                    | Description  |
|-------------------------|--|
| <a href="#">BaseDir</a> | Used to set the base path for MySQL Embedded server.                     |
| <a href="#">DataDir</a> | Used to set the path to the data directory.                              |
| <a href="#">Params</a>  | Used to specify the list of command line parameters for Embedded server. |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Used to set the base path for MySQL Embedded server.

## Class

[TMyEmbConnection](#)

## Syntax

```
property BaseDir: string stored False;
```

## Remarks

Use the BaseDir property to set the base path for MySQL Embedded server. All paths are usually resolved relative to this. Corresponds to --basedir parameter. See MySQL Reference Manual for detailed description.  
The default value is '.' and all auxiliary files must be located in the same folder with the executable file of the project.

## See Also

- [DataDir](#)
- [Params](#)
- [Embedded Server](#)

---

© 1997-2012 Devart. All Rights Reserved.

Used to set the path to the data directory.

## Class

[TMyEmbConnection](#)

## Syntax

```
property DataDir: string stored False;
```

## Remarks

Use the DataDir property to set the path to the data directory. Corresponds to --datadir parameter. See MySQL Reference Manual for a detailed description.

The default value is 'data' and all data files must be in the subfolder data by the path specified in BaseDir.

It is convenient to use this property for cases when the program is started from ReadOnly data carrier (CD-ROM, network etc). Also its usage can be suitable to separate data of different users.

## See Also

- [BaseDir](#)
- [Params](#)
- [Embedded Server](#)

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify the list of command line parameters for Embedded server.

## Class

[TMyEmbConnection](#)

## Syntax

```
property Params: TStrings;
```

## Remarks

Use the Params property to specify the list of command line parameters for Embedded server.

Pay attention that all paths must be set through "/" but not "\".

Params property also stores values for BaseDir and DataDir.

If no parameters are set in Params property its value will be obtained from the file of configuration of EmbServer (my.ini or my.cnf). Please see MySQL Reference for details.

libmysqld library is reloaded only on changing parameters.

Note, parameters names are case-sensitive.

## See Also

- [BaseDir](#)
- [DataDir](#)
- [Embedded Server](#)

---

© 1997-2012 Devart. All Rights Reserved.

Events of the **TMyEmbConnection** class.

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

## Public

| Name   | Description  |
|--|--|
| <a href="#">ApplyUpdates</a> (inherited from <a href="#">TCustomDAConnection</a> )       | Overloaded. Applies changes in datasets.   |
| <a href="#">AssignConnect</a> (inherited from <a href="#">TCustomMyConnection</a> )      | Shares database connection between the TCustomMyConnection components.   |
| <a href="#">ClientVersion</a> (inherited from <a href="#">TCustomMyConnection</a> )      | Contains the version of the MySQL Client library.  |
| <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="#">ConnectionTimeout</a> (inherited from <a href="#">TCustomMyConnection</a> )  | Used to specify the amount of time to attempt to establish a connection.   |
| <a href="#">ConvertEOL</a> (inherited from <a href="#">TCustomDAConnection</a> )         | Allows customi ing line breaks in string fields and parameters.  |
| <a href="#">CreateDataSet</a> (inherited from <a href="#">TCustomMyConnection</a> )      | Returns a new instance of TCustomMyDataSet class and associates it with this connection object.                    |
| <a href="#">CreateSQL</a> (inherited from <a href="#">TCustomDAConnection</a> )          | Creates a component for queries execution.   |
| <a href="#">Database</a> (inherited from <a href="#">TCustomMyConnection</a> )           | Used to specify a database name that is a default source of data for SQL queries once a connection is established. |
| <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="#">TCustomMyConnection</a> )            | Executes any SQL statement outside <a href="#">TMyQuery</a> or <a href="#">TMyCommand</a> components.              |
| <a href="#">ExecSQLEx</a> (inherited from <a href="#">TCustomDAConnection</a> )          | Executes any SQL statement outside the TQuery or TSQL components.  |
| <a href="#">GetCharsetNames</a> (inherited from <a href="#">TCustomMyConnection</a> )    | Populates a string list with the names of available charsets.  |
| <a href="#">GetDatabaseNames</a> (inherited from <a href="#">TCustomDAConnection</a> )   | Returns a database list from the server.   |
| <a href="#">GetExecuteInfo</a> (inherited from <a href="#">TCustomMyConnection</a> )     | Returns the result of the last query execution.  |
| <a href="#">GetStoredProcNames</a> (inherited from <a href="#">TCustomDAConnection</a> ) | Returns a list of stored procedures from the server.   |
| <a href="#">GetTriggerNames</a> (inherited from <a href="#">TCustomMyConnection</a> )    | Returns a list of triggers from the server.  |
| <a href="#">InTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )      | Indicates whether the transaction is active.   |

|   |  |
|---|--|
| <a href="#">IsolationLevel</a> (inherited from <a href="#">TCustomMyConnection</a> )      | Used to specify extent to which all outside transactions interfere with subsequent transactions of current connection. |
| <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="#">Options</a> (inherited from <a href="#">TCustomMyConnection</a> )             | Specifies the behaviour of the TMyConnectionOptions object.  |
| <a href="#">Password</a> (inherited from <a href="#">TCustomDAConnection</a> )            | Serves to supply a password for login.   |
| <a href="#">Ping</a> (inherited from <a href="#">TCustomMyConnection</a> )                | Allows to avoid automatic disconnection of the client by the server.   |
| <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="#">ReleaseSavepoint</a> (inherited from <a href="#">TCustomMyConnection</a> )    | Releases the specified savepoint without affecting any work that has been performed after its creation.                |
| <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="#">RollbackToSavepoint</a> (inherited from <a href="#">TCustomMyConnection</a> ) | Cancels all updates for the current transaction.   |
| <a href="#">Savepoint</a> (inherited from <a href="#">TCustomMyConnection</a> )           | Defines a point in the transaction to which you can roll back later.   |
| <a href="#">Server</a> (inherited from <a href="#">TCustomDAConnection</a> )              | Serves to supply the server name for login.  |
| <a href="#">ServerVersion</a> (inherited from <a href="#">TCustomMyConnection</a> )       | Holds the version of MySQL server.   |
| <a href="#">StartTransaction</a> (inherited from <a href="#">TCustomDAConnection</a> )    | Begins a new user transaction.   |
| <a href="#">ThreadId</a> (inherited from <a href="#">TCustomMyConnection</a> )            | Used to return the thread ID of the current connection.  |
| <a href="#">Username</a> (inherited from <a href="#">TCustomDAConnection</a> )            | Used to supply a user name for login.  |

## Published

| Name                       | Description  |
|----------------------------|--|
| <a href="#">OnLog</a>      | Occurs when MySQL server writes down to General Query Log. |
| <a href="#">OnLogError</a> | Occurs when MySQL server writes down to Error Log.         |

## See Also

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

Occurs when MySQL server writes down to General Query Log.

### Class

[TMyEmbConnection](#)

### Syntax

```
property OnLog: TMyLogEvent;
```

### Remarks

The OnLog event occurs when MySQL server writes down to General Query Log, the same as on using --log option. See MySQL Reference Manual for detailed description.

On assigning handler for OnLog event MySQL server does not make output to common log-file. This event is available only for Win32.

### See Also

- [OnLogError](#)
  - [Embedded Server](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Occurs when MySQL server writes down to Error Log.

### Class

[TMyEmbConnection](#)

### Syntax

```
property OnLogError: TMyLogEvent;
```

### Remarks

The OnLogError event occurs when MySQL server writes down to Error Log, the same as on using --log-error option. See MySQL Reference Manual for detailed description. It is convenient to use OnLogError event to search errors in the server configuration, as often after error message is displayed an application is terminated.

On assigning handler for OnLogError event MySQL server does not make output to common error log-file.

This event is available only for Win32.

### See Also

- [OnLog](#)
  - [Embedded Server](#)
- 

© 1997-2012 Devart. All Rights Reserved.

## 17.23 MyLoader

This unit contains implementation of the TMyLoader component.

### Classes

| Name                      | Description  |
|---------------------------|--|
| <a href="#">TMyColumn</a> | A component representing the attributes for column loading.      |
| <a href="#">TMyLoader</a> | TMyLoader allows to load external data into the server database. |

### Types

| Name                             | Description   |
|----------------------------------|---|
| <a href="#">TMyLoaderOptions</a> | Represents the set of <a href="#">TMyLoaderOption</a> . |

### Enumerations

| Name                             | Description  |
|----------------------------------|--|
| <a href="#">TMyDuplicateKeys</a> | Specifies the way conflicts with duplicated key values will be resolved. |
| <a href="#">TMyLoaderOption</a>  | Specifies the behaviour of a TMyLoader object.                           |

## 17.23.1 Classes

Classes in the **MyLoader** unit.

### Classes

| Name                      | Description  |
|---------------------------|--|
| <a href="#">TMyColumn</a> | A component representing the attributes for column loading.      |
| <a href="#">TMyLoader</a> | TMyLoader allows to load external data into the server database. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.23.1.1 MyLoader.TMyColumn Class

A component representing the attributes for column loading.

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

### Unit

[MyLoader](#)

### Syntax

```
TMyColumn = class(TDAColumn) ;
```

### Remarks

Each [TMyLoader](#) uses [TDAColumns](#) to maintain a collection of TMSColumn objects. TMyColumn object represents the attributes for column loading. Every TMyColumn object corresponds to one of the table fields with the same name as its [TDAColumn.Name](#) property.

To create columns at design-time use column editor of the TMyLoader component.

### Inheritance Hierarchy

TObject

[TDAColumn](#)

**TMyColumn**

### See Also

- [TMyLoader](#)
- [TDAColumns](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyColumn](#) class overview.

### Properties

| Name  | Description  |
|---|--|
| <a href="#">FieldType</a> (inherited from <a href="#">TDAColumn</a> ) | Used to specify the types of values that will be loaded. |
| <a href="#">Name</a> (inherited from <a href="#">TDAColumn</a> )      | Used to specify the field name of loading table.         |

© 1997-2012 Devart. All Rights Reserved.

#### 17.23.1.2 MyLoader.TMyLoader Class

TMyLoader allows to load external data into the server database.

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

### Unit

[MyLoader](#)



## Syntax

```
TMyLoader = class (TDALoader) ;
```

## Remarks

TMyLoader serves for fast loading of data to the server.

TMyLoader work is based on generation INSERT statements which insert data by several rows for one time (see [TMyLoader.RowsPerQuery](#)).

This class is used to simplify INSERT statements generation.

To affect on performance set MyLoader.Connection.Options.Compression property to True.

The quicker way is to generate text file on the client side and load it using [TMyBackup.Restore](#) method with Mode = bmText.

## Inheritance Hierarchy

```

TObject
  TDALoader
    TMyLoader
  
```

## See Also

- [TMyBackup](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyLoader](#) class overview.

## Properties

| 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>  | Used to specify a connection object that will be used to connect to a data store.   |
| <a href="#">DuplicateKeys</a>   | Used to specify in what way conflicts with duplicated key values will be resolved.  |
| <a href="#">Options</a>   | Specifies the behaviour of TMyLoader object.  |
| <a href="#">RowsPerQuery</a>  | Used to get or set the number of rows that will be send to the server for one time. |
| <a href="#">TableName</a> (inherited from <a href="#">TDALoader</a> ) | Used to specify the name of the table to which data will be loaded.                 |

## 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> (inherited from <a href="#">TDALoader</a> ) | Occurs when it is needed to put column values. |

[OnProgress](#) (inherited from [TDALoader](#))

Occurs if handling data loading progress of the [TDALoader.LoadFromDataSet](#) method is needed.

[OnPutData](#) (inherited from [TDALoader](#))

Occurs when putting loading data by rows is needed.

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyLoader** class.

For a complete list of the **TMyLoader** class members, see the [TMyLoader 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="#">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.   |
| <a href="#">TableName</a> (inherited from <a href="#">TDALoader</a> )       | Used to specify the name of the table to which data will be loaded.   |

## Published

| Name                          | Description   |
|-------------------------------|---|
| <a href="#">Connection</a>    | Used to specify a connection object that will be used to connect to a data store.   |
| <a href="#">DuplicateKeys</a> | Used to specify in what way conflicts with duplicated key values will be resolved.  |
| <a href="#">Options</a>       | Specifies the behaviour of TMyLoader object.  |
| <a href="#">RowsPerQuery</a>  | Used to get or set the number of rows that will be send to the server for one time. |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Used to specify a connection object that will be used to connect to a data store.

## Class

[TMyLoader](#)

### Syntax

**property** Connection: [TCustomMyConnection](#);

### 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 TMyConnection objects. At runtime, set the Connection property to reference an existing TMyConnection object.

### See Also

- [TCustomMyConnection](#)

---

© 1997-2012 Devart. All Rights Reserved.

Used to specify in what way conflicts with duplicated key values will be resolved.

### Class

[TMyLoader](#)

### Syntax

**property** DuplicateKeys: [TMyDuplicateKeys](#) **default** dkNone;

### Remarks

Use the DuplicateKeys property to specify in what way conflicts with duplicated key values will be resolved.

---

© 1997-2012 Devart. All Rights Reserved.

Specifies the behaviour of TMyLoader object.

### Class

[TMyLoader](#)

### Syntax

**property** Options: [TMyLoaderOptions](#) **default** [];

### Remarks

Set the properties of Options to specify the behaviour of a TMyLoader object.

---

© 1997-2012 Devart. All Rights Reserved.

Used to get or set the number of rows that will be send to the server for one time.

### Class

[TMyLoader](#)

### Syntax

**property** RowsPerQuery: integer **default** 0;

### Remarks

Use the RowsPerQuery property to get or set the number of rows that will be send to the server for one time. The default value is 0. In this case rows will be grouped by 16Kb (the default value of net buffer length).

---

© 1997-2012 Devart. All Rights Reserved.

## 17.23.2 Types

Types in the **MyLoader** unit.

### Types

| Name                             | Description   |
|----------------------------------|---|
| <a href="#">TMyLoaderOptions</a> | Represents the set of <a href="#">TMyLoaderOption</a> . |

---

© 1997-2012 Devart. All Rights Reserved.

#### 17.23.2.1 MyLoader.TMyLoaderOptions Set

Represents the set of [TMyLoaderOption](#).

### Unit

[MyLoader](#)

### Syntax

```
TMyLoaderOptions = set of TMyLoaderOption;
```

---

© 1997-2012 Devart. All Rights Reserved.

### 17.23.3 Enumerations

Enumerations in the **MyLoader** unit.

#### Enumerations

| Name                             | Description  |
|----------------------------------|--|
| <a href="#">TMyDuplicateKeys</a> | Specifies the way conflicts with duplicated key values will be resolved. |
| <a href="#">TMyLoaderOption</a>  | Specifies the behaviour of a TMyLoader object.                           |

© 1997-2012 Devart. All Rights Reserved.

#### 17.23.3.1 MyLoader.TMyDuplicateKeys Enumeration

Specifies the way conflicts with duplicated key values will be resolved.

##### Unit

[MyLoader](#)

##### Syntax

```
TMyDuplicateKeys = (dkNone, dkIgnore, dkReplace);
```

##### Values

| Value            | Meaning  |
|------------------|--|
| <b>dkIgnore</b>  | The new record with duplicated key value will be ignored silently.   |
| <b>dkNone</b>    | An error will be raised and loading process will be stopped after an attempt to insert a record with already existing key value. |
| <b>dkReplace</b> | The old record in database with duplicated key values will be silently replaced with the new one.                                |

© 1997-2012 Devart. All Rights Reserved.

#### 17.23.3.2 MyLoader.TMyLoaderOption Enumeration

Specifies the behaviour of a TMyLoader object.

##### Unit

[MyLoader](#)

##### Syntax

```
TMyLoaderOption = (loLock, loDelayed);
```

##### Values

| Value            | Meaning                            |
|------------------|------------------------------------|
| <b>loDelayed</b> | Uses INSERT DELAYED syntax.        |
| <b>loLock</b>    | Locks tables while inserting data. |

© 1997-2012 Devart. All Rights Reserved.

## **17.24 MyScript**

This unit contains implementation of the TMyScript component.

### **Classes**

| <b>Name</b>               | <b>Description</b>   |
|---------------------------|--|
| <a href="#">TMyScript</a> | A component for executing several SQL statements one by one. |

---

© 1997-2012 Devart. All Rights Reserved.

## 17.24.1 Classes

Classes in the **MyScript** unit.

### Classes

| Name                      | Description  |
|---------------------------|--|
| <a href="#">TMyScript</a> | A component for executing several SQL statements one by one. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.24.1.1 MyScript.TMyScript Class

A component for executing several SQL statements one by one.

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

### Unit

[MyScript](#)

### Syntax

```
TMyScript = 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 [TMyCommand](#). Usually it is not a good solution. With only one TMyScript component you can execute several SQL statements as one. This sequence of statements is named script. To separate single statements use semicolon (;), slash (/), and DELIMITER . 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 TMyScript shows exception and continues execution.

If you need to create several Stored Procedures (Functions) at a single script, use slash (/) to separate commands for creating stored procedures.

### Inheritance Hierarchy

TObject  
    [TDAScript](#)  
        **TMyScript**

### See Also

- [TMyCommand](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyScript](#) 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="#">DataSet</a>   | Used to retrieve the results of SELECT statements execution inside a script.      |
| <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.                   |

[EndOffset](#) (inherited from [TDAScript](#))

[EndPos](#) (inherited from [TDAScript](#))

[Macros](#) (inherited from [TDAScript](#))

[SQL](#) (inherited from [TDAScript](#))

[StartLine](#) (inherited from [TDAScript](#))

[StartOffset](#) (inherited from [TDAScript](#))

[StartPos](#) (inherited from [TDAScript](#))

[Statements](#) (inherited from [TDAScript](#))

[UseOptimi ation](#)

Used to get the offset in the last line of the current statement.

Used to get the end position of the current statement.

Used to change SQL script text in design- or run-time easily.

Used to get or set script text.

Used to get the current statement start line number in a script.

Used to get the offset in the first line of the current statement.

Used to get the start position of the current statement in a script.

Contains a list of statements obtained from the SQL property.

Used to unit small queries into blocks for block executing if possible.

## Methods

| Name  | Description  |
|---|--|
| <a href="#">BreakExec</a> (inherited from <a href="#">TDAScript</a> )     | Stops script execution.  |
| <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.  |

## Events

| Name  | Description  |
|---|--|
| <a href="#">AfterExecute</a> (inherited from <a href="#">TDAScript</a> )  | Occurs after a SQL script execution.   |
| <a href="#">BeforeExecute</a> (inherited from <a href="#">TDAScript</a> ) | Occurs when taking a specific action before executing the current SQL statement is needed. |
| <a href="#">OnError</a> (inherited from <a href="#">TDAScript</a> )       | Occurs when MySQL raises an error.   |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyScript** class.

For a complete list of the **TMyScript** class members, see the [TMyScript 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="#">TDAAScript</a> )        | Used to get the end position of the current statement.                             |
| <a href="#">ErrorOffset</a> (inherited from <a href="#">TDAAScript</a> )   | Used to get the offset of the statement if the Execute method raised an exception. |
| <a href="#">Execute</a> (inherited from <a href="#">TDAAScript</a> )       | Executes a script.   |
| <a href="#">ExecuteFile</a> (inherited from <a href="#">TDAAScript</a> )   | Executes SQL statements contained in a file.                                       |
| <a href="#">ExecuteNext</a> (inherited from <a href="#">TDAAScript</a> )   | Executes the next statement in the script and then stops.                          |
| <a href="#">ExecuteStream</a> (inherited from <a href="#">TDAAScript</a> ) | Executes SQL statements contained in a stream object.                              |
| <a href="#">FindMacro</a> (inherited from <a href="#">TDAAScript</a> )     | Indicates whether a specified macro exists in a dataset.                           |
| <a href="#">MacroByName</a> (inherited from <a href="#">TDAAScript</a> )   | Finds a Macro with the name passed in Name.  |
| <a href="#">StartLine</a> (inherited from <a href="#">TDAAScript</a> )     | Used to get the current statement start line number in a script.                   |
| <a href="#">StartOffset</a> (inherited from <a href="#">TDAAScript</a> )   | Used to get the offset in the first line of the current statement.                 |
| <a href="#">StartPos</a> (inherited from <a href="#">TDAAScript</a> )      | Used to get the start position of the current statement in a script.               |
| <a href="#">Statements</a> (inherited from <a href="#">TDAAScript</a> )    | Contains a list of statements obtained from the SQL property.                      |

## Published

| Name   | Description  |
|--|--|
| <a href="#">AfterExecute</a> (inherited from <a href="#">TDAAScript</a> )  | Occurs after a SQL script execution.   |
| <a href="#">BeforeExecute</a> (inherited from <a href="#">TDAAScript</a> ) | Occurs when taking a specific action before executing the current SQL statement is needed. |
| <a href="#">Connection</a>   | Used to specify a connection object that will be used to connect to a data store.          |
| <a href="#">DataSet</a>  | Used to retrieve the results of SELECT statements execution inside a script.               |
| <a href="#">Debug</a> (inherited from <a href="#">TDAAScript</a> )         | Used to display the script execution and all its parameter values.                         |
| <a href="#">Delimiter</a> (inherited from <a href="#">TDAAScript</a> )     | Used to set the delimiter string that separates script statements.                         |
| <a href="#">Macros</a> (inherited from <a href="#">TDAAScript</a> )        | Used to change SQL script text in design- or run-time easily.                              |
| <a href="#">OnError</a> (inherited from <a href="#">TDAAScript</a> )       | Occurs when MySQL raises an error.   |
| <a href="#">SQL</a> (inherited from <a href="#">TDAAScript</a> )           | Used to get or set script text.  |
| <a href="#">UseOptimi ation</a>  | Used to unit small queries into blocks for block executing if possible.                    |

## See Also

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

© 1997-2012 Devart. All Rights Reserved.

Used to specify a connection object that will be used to connect to a data store.

## Class

[TMyScript](#)

### Syntax

**property** Connection: [TCustomMyConnection](#);

### 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 TMyConnection objects. At run-time, set Connection property to reference an existing TMyConnection object.

### See Also

- [TCustomMyConnection](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to retrieve the results of SELECT statements execution inside a script.

### Class

[TMyScript](#)

### Syntax

**property** DataSet: [TCustomMyDataSet](#);

### Remarks

Use the DataSet property if you need to retrieve the results of SELECT statements execution inside a script.

### See Also

- [TDA Script.Execute](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Used to unit small queries into blocks for block executing if possible.

### Class

[TMyScript](#)

### Syntax

**property** UseOptimization: boolean;

### Remarks

Use the UseOptimization property to unit small queries into blocks for block executing if possible. The UseOptimization option does not affect the [TDA Script.ExecuteNext](#) method performance. It works only for the [TDA Script.Execute](#) method.

---

© 1997-2012 Devart. All Rights Reserved.

## **17.25 MyServerControl**

This unit contains implementation of the TMyServerControl component.

### **Classes**

| <b>Name</b>                      | <b>Description</b>   |
|----------------------------------|--|
| <a href="#">TMyServerControl</a> | A component for server control and standard service tasks execution. |

© 1997-2012 Devart. All Rights Reserved.

## 17.25.1 Classes

Classes in the **MyServerControl** unit.

### Classes

| Name                             | Description  |
|----------------------------------|--|
| <a href="#">TMyServerControl</a> | A component for server control and standard service tasks execution. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.25.1.1 MyServerControl.TMyServerControl Class

A component for server control and standard service tasks execution.

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

### Unit

[MyServerControl](#)

### Syntax

```
TMyServerControl = class (TCustomMyDataSet);
```

### Remarks

TMyServerControl is a direct descendant of [TCustomMyDataSet](#) component. It serves to control the server and execute standard service tasks. List of the functions can be divided in the following groups:

- Create and delete a database ( [TMyServerControl.CreateDatabase](#), [TMyServerControl.DropDatabase](#) );
- Serves to manage MySQL services. available only for Windows ( [TMyServerControl.ServiceStart](#), [TMyServerControl.ServiceStop](#), [TMyServerControl.GetServiceNames](#), [TMyServerControl.ServiceStatus](#) );
- Manage MySQL server system variables ( [TMyServerControl.Variables](#), [TMyServerControl.ShowVariables](#) );
- Flush server data ( [TMyServerControl.Flush](#) );
- Manage tables ( [TMyServerControl.AnalyzeTable](#), [TMyServerControl.OptimizeTable](#), [TMyServerControl.CheckTable](#), [TMyServerControl.RepairTable](#) );
- Manage processes ( [TMyServerControl.ShowProcessList](#), [TMyServerControl.KillProcess](#) );
- Obtain current information about the server ( [TMyServerControl.ShowStatus](#) ).

The last three groups show report as DataSet.

### Inheritance Hierarchy

```

TObject
  TMemDataSet
    TCustomDADataset
      TCustomMyDataSet
        TMyServerControl
  
```

### See Also

- [TMyQuery](#)

© 1997-2012 Devart. All Rights Reserved.

[TMyServerControl](#) class overview.

### Properties

| Name  | Description   |
|---|---|
| <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="#"><u>CommandTimeout</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> ) | Used to specify the amount of time to attempt execution of a command.  |
| <a href="#"><u>Connection</u></a>   | Used to specify a connection object to use for connecting to a data store.   |
| <a href="#"><u>Debug</u></a>  | Used to display executing statement.   |
| <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>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>FetchAll</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )       | Description is not available at the moment.  |
| <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>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>InsertId</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )       | Returns the ID generated for an AUTO INCREMENT column by the previous query.   |
| <a href="#"><u>IsQuery</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )        | Used to check whether 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>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>TCustomMyDataSet</u></a> )       | Specifies when to perform locking of an 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="#">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="#">TCustomMyDataSet</a> )        | Specifies the behaviour of TCustomMyDataSet 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="#">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="#">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="#">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="#">TableNames</a>  | Holds a list of tables that will be processed by the <a href="#">TMyServerControl.AnalyzeTable</a> , <a href="#">TMyServerControl.OptimizeTable</a> , <a href="#">TMyServerControl.CheckTable</a> , <a href="#">TMyServerControl.RepairTable</a> methods. |
| <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.  |

[Variables](#)

Used to get or set the values of MySQL Server system variables.

**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="#">Analy_eTable</a>  | Analy es and stores the key distribution for the table   |
| <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="#">CheckTable</a>  | Checks MyISAM and InnoDB tables for failures.  |
| <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="#">CreateDatabase</a>  | Creates a new database with the specified name.  |
| <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="#">DropDatabase</a>  | Drops the specified database.  |
| <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="#">TCustomDADataset</a> )          | Used to learn whether 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="#">TCustomDADataset</a> )        | Determines if a parameter with the specified name exists in a dataset.   |
| <a href="#">Flush</a>   | Flushes server data from the memory to disk.   |

|  |  |
|--|--|
| <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="#">GetFieldEnum</a> (inherited from <a href="#">TCustomMyDataSet</a> )      | Retrieve the list of acceptable values for a specified field given by the FieldName parameter.                             |
| <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="#">GetServiceNames</a>  | Populates a list of MySQL services at the server to which Connection is connected.   |
| <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="#">KillProcess</a>  | Breaks the execution of the specified process.   |
| <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="#">TCustomMyDataSet</a> )              | Overloaded. Locks the current record for the current connection.   |
| <a href="#">LockTable</a> (inherited from <a href="#">TCustomMyDataSet</a> )         | Locks table for the current connection.  |
| <a href="#">MacroByName</a> (inherited from <a href="#">TCustomDADataset</a> )       | Finds a Macro with the name passed in Name.  |
| <a href="#">OptimizeTable</a>  | Packs tables deleting unused places from the file at the disk and defragmenting it.  |
| <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="#">Prepare</a> (inherited from <a href="#">TCustomDADataset</a> )           | Allocates, opens, and parses cursor for a query.   |
| <a href="#">RefreshQuick</a> (inherited from <a href="#">TCustomMyDataSet</a> )      | Retrieves changes posted to the server by another clients on the client side quickly.                                      |
| <a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )     | Actualizes field values for the current record.  |
| <a href="#">RepairTable</a>  | Repairs specified tables.  |
| <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.  |



[Resync](#) (inherited from [TCustomDADataset](#))

[RevertRecord](#) (inherited from [TMemDataSet](#))

[SaveSQL](#) (inherited from [TCustomDADataset](#))

[SaveToXML](#) (inherited from [TMemDataSet](#))

[ServiceStart](#)

[ServiceStatus](#)

[ServiceStop](#)

[SetOrderBy](#) (inherited from [TCustomDADataset](#))

[ShowProcessList](#)

[ShowStatus](#)

[ShowVariables](#)

[SQLSaved](#) (inherited from [TCustomDADataset](#))

[UnLock](#) (inherited from [TCustomDADataset](#))

[UnLockTable](#) (inherited from [TCustomMyDataSet](#))

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

[UpdateStatus](#) (inherited from [TMemDataSet](#))

Resynchronizes the dataset with underlying physical data when making calls that may change the internal cursor position.

Cancels changes made to the current record when cached updates are enabled.

Saves the SQL property value to BaseSQL.

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Starts the specified service at the server to which Connection is connected.

Returns a status of the specified service at the server to which Connection is connected.

Stops the specified service at the server to which Connection is connected.

Builds an ORDER BY clause of a SELECT statement.

Shows the list of the processes.

Shows the current state of the server.

Shows the values of some MySQL system variables that are in effect for the current connection.

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

Releases a record lock.

Releases a table locked by the [TCustomMyDataSet.LockTable](#) method.

Frees the resources allocated for a previously prepared query on the server and client sides.

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.

## 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.                     |

© 1997-2012 Devart. All Rights Reserved.

Properties of the **TMyServerControl** class.

For a complete list of the **TMyServerControl** class members, see the [TMyServerControl 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="#">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="#">CommandTimeout</a> (inherited from <a href="#">TCustomMyDataSet</a> )      | Used to specify the amount of time to attempt execution of a command.  |
| <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="#">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>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>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>FetchAll</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )          | Description is not available at the moment.  |
| <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="#"><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>TCustomDADataset</u></a> )         | Determines if a parameter with the specified name exists in a dataset.   |
| <a href="#"><u>GetBlob</u></a> (inherited from <a href="#"><u>TMemDataSet</u></a> )                | Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself 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>GetFieldEnum</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Retrieve the list of acceptable values for a specified field given by the FieldName parameter.   |
| <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="#">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="#">InsertId</a> (inherited from <a href="#">TCustomMyDataSet</a> )     | Returns the ID generated for an AUTO INCREMENT column by the previous query.   |
| <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="#">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="#">TCustomMyDataSet</a> )         | Overloaded. Locks the current record for the current connection.   |
| <a href="#">LockMode</a> (inherited from <a href="#">TCustomMyDataSet</a> )     | Specifies when to perform locking of an editing record.  |
| <a href="#">LockTable</a> (inherited from <a href="#">TCustomMyDataSet</a> )    | Locks table for the current connection.  |
| <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> (inherited from <a href="#">TCustomMyDataSet</a> )      | Specifies the behaviour of TCustomMyDataSet object.  |

|   |  |
|---|--|
| <a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )    | Sets or uses 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>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>RefreshQuick</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )   | Retrieves changes posted to the server by another clients on the client side quickly.                                      |
| <a href="#"><u>RefreshRecord</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )  | Actualizes 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="#"><u>Resync</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )         | Resynchronizes the dataset with underlying physical data when making calls that may change the internal cursor position.   |
| <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>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>SaveSQL</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )        | Saves the SQL property value to BaseSQL.   |
| <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>SetOrderBy</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )     | Builds an ORDER BY clause of a SELECT statement.   |
| <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.                               |

[SQLLock](#) (inherited from [TCustomDADataset](#))

[SQLRefresh](#) (inherited from [TCustomDADataset](#))

[SQLSaved](#) (inherited from [TCustomDADataset](#))

[SQLUpdate](#) (inherited from [TCustomDADataset](#))

[UniDirectional](#) (inherited from [TCustomDADataset](#))

[UnLock](#) (inherited from [TCustomDADataset](#))

[UnLockTable](#) (inherited from [TCustomMyDataSet](#))

[UnPrepare](#) (inherited from [TMemDataSet](#))

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

[UpdateResult](#) (inherited from [TMemDataSet](#))

[UpdatesPending](#) (inherited from [TMemDataSet](#))

[UpdateStatus](#) (inherited from [TMemDataSet](#))

[Variables](#)

Used to specify a SQL statement that will be used to perform a record lock.

Used to specify a SQL statement that will be used to refresh current record by calling the [TCustomDADataset.RefreshRecord](#) procedure.

Determines if the [SQL](#) property value was saved to the [BaseSQL](#) property.

Used to specify a SQL statement that will be used when applying an update to a dataset.

Used if an application does not need bidirectional access to records in the result set.

Releases a record lock.

Releases a table locked by the [TCustomMyDataSet.LockTable](#) method.

Frees the resources allocated for a previously prepared query on the server and client sides.

Used to indicate the update status for the current record when cached updates are enabled.

Reads the status of the latest call to the [ApplyUpdates](#) method while cached updates are enabled.

Used to check the status of the cached updates buffer.

Indicates the current update status for the dataset when cached updates are enabled.

Used to get or set the values of MySQL Server system variables.

## Published

| Name                       | Description   |
|----------------------------|---|
| <a href="#">Connection</a> | Used to specify a connection object to use for connecting to a data store.  |
| <a href="#">Debug</a>      | Used to display executing statement.  |
| <a href="#">TableNames</a> | Holds a list of tables that will be processed by the <a href="#">TMyServerControl.AnalyzeTable</a> , <a href="#">TMyServerControl.OptimizeTable</a> , <a href="#">TMyServerControl.CheckTable</a> , <a href="#">TMyServerControl.RepairTable</a> methods. |

## See Also

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

Used to specify a connection object to use for connecting to a data store.

### Class

[TMyServerControl](#)

### Syntax

**property** Connection: [TCustomMyConnection](#);

### 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 TMyConnection objects. At run-time, set Connection property to reference an existing TMyConnection object.

### See Also

- [TCustomMyConnection](#)

---

© 1997-2012 Devart. All Rights Reserved.

Used to display executing statement.

### Class

[TMyServerControl](#)

### Syntax

**property** Debug: boolean;

### Remarks

Set the Debug property to True to display executing statement. The default value is False.

**Note:** To enable debug form display you should explicitly include MyDacVcl (MyDacClx under Kylix) unit into your project.

### See Also

- [TCustomDADataset.Debug](#)

---

© 1997-2012 Devart. All Rights Reserved.

Holds a list of tables that will be processed by the [Analy eTable](#), [Optimi eTable](#), [CheckTable](#), [RepairTable](#) methods.

### Class

[TMyServerControl](#)

### Syntax

**property** TableNames: string;

### Remarks

Contains a list of tables that will be processed by the [Analy eTable](#), [Optimi eTable](#), [CheckTable](#), [RepairTable](#) methods.

Table names are separated by comma or semicolon. If it has an empty value, all tables presented in the database will be processed.

### See Also

- [Analy eTable](#)
- [Optimi eTable](#)

- [CheckTable](#)
- [RepairTable](#)
- `M:Devart.Dac.TCustomDAConnection.GetTableNames(Borland.Vcl.TStrings,System.Boolean)`

© 1997-2012 Devart. All Rights Reserved.

Used to get or set the values of MySQL Server system variables.

## Class

[TMyServerControl](#)

## Syntax

**property** Variables[**const** VarName: string]: string;

### Parameters

*VarName*

Holds the name of MySQL server system variables.

## Remarks

Use the Variables property to get or set the values of MySQL Server system variables. String values should be assigned with quotes.

## Example

For example:

```
MyServerControl1.Variables['max_allowed_packet'] := '1234567890';
but:
MyServerControl.Variables['time_format'] := '%H:%i:%s';
```

© 1997-2012 Devart. All Rights Reserved.

Methods of the **TMyServerControl** class.

For a complete list of the **TMyServerControl** class members, see the [TMyServerControl 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="#">Analy_eTable</a>  | Analy es and stores the key distribution for the table  |
| <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="#"><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>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>CheckTable</u></a>  | Checks MyISAM and InnoDB tables for failures.  |
| <a href="#"><u>CommandTimeout</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Used to specify the amount of time to attempt execution of a command.  |
| <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>TCustomMyDataSet</u></a> )          | Used to specify a connection object that will be used to connect to a data store.  |
| <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>CreateDatabase</u></a>  | Creates a new database with the specified name.  |
| <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>DropDatabase</u></a>  | Drops the specified database.  |
| <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>FetchAll</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )            | Description is not available at the moment.  |
| <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.   |

[FilterSQL](#) (inherited from [TCustomDADataset](#))

[FinalSQL](#) (inherited from [TCustomDADataset](#))

[FindKey](#) (inherited from [TCustomDADataset](#))

[FindMacro](#) (inherited from [TCustomDADataset](#))

[FindNearest](#) (inherited from [TCustomDADataset](#))

[FindParam](#) (inherited from [TCustomDADataset](#))

[Flush](#)

[GetBlob](#) (inherited from [TMemDataSet](#))

[GetDataType](#) (inherited from [TCustomDADataset](#))

[GetFieldEnum](#) (inherited from [TCustomMyDataSet](#))

[GetFieldObject](#) (inherited from [TCustomDADataset](#))

[GetFieldPrecision](#) (inherited from [TCustomDADataset](#))

[GetFieldScale](#) (inherited from [TCustomDADataset](#))

[GetOrderBy](#) (inherited from [TCustomDADataset](#))

[GetServiceNames](#)

[GotoCurrent](#) (inherited from [TCustomDADataset](#))

[IndexFieldNames](#) (inherited from [TMemDataSet](#))

[InsertId](#) (inherited from [TCustomMyDataSet](#))

[IsQuery](#) (inherited from [TCustomDADataset](#))

[KeyFields](#) (inherited from [TCustomDADataset](#))

[KillProcess](#)

Used to change the WHERE clause of SELECT statement and reopen a query.

Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.

Searches for a record which contains specified field values.

Indicates whether a specified macro exists in a dataset.

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.

Determines if a parameter with the specified name exists in a dataset.

Flushes server data from the memory to disk.

Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.

Returns internal field types defined in the MemData and accompanying modules.

Retrieve the list of acceptable values for a specified field given by the FieldName parameter.

Returns a multireference shared object from field.

Retrieves the precision of a number field.

Retrieves the scale of a number field.

Retrieves an ORDER BY clause from a SQL statement.

Populates a list of MySQL services at the server to which Connection is connected.

Sets the current record in this dataset similar to the current record in another dataset.

Used to get or set the list of fields on which the recordset is sorted.

Returns the ID generated for an AUTO INCREMENT column by the previous query.

Used to check whether SQL statement returns rows.

Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.

Breaks the execution of the specified process.

|   |  |
|---|--|
| <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>TCustomMyDataSet</u></a> )         | Overloaded. Locks the current record for the current connection.   |
| <a href="#"><u>LockMode</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )     | Specifies when to perform locking of an editing record.  |
| <a href="#"><u>LockTable</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )    | Locks table for the current connection.  |
| <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>OptimizeTable</u></a>  | Packs tables deleting unused places from the file at the disk and defragmenting it.  |
| <a href="#"><u>Options</u></a> (inherited from <a href="#"><u>TCustomMyDataSet</u></a> )      | Specifies the behaviour of TCustomMyDataSet object.  |
| <a href="#"><u>ParamByName</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></a> )  | Sets or uses 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>Prepare</u></a> (inherited from <a href="#"><u>TCustomDADataset</u></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="#">RefreshQuick</a> (inherited from <a href="#">TCustomMyDataSet</a> )   | Retrieves changes posted to the server by another clients on the client side quickly.                                   |
| <a href="#">RefreshRecord</a> (inherited from <a href="#">TCustomDADataset</a> )  | Actuali es field values for the current record.   |
| <a href="#">RepairTable</a>   | Repairs specified tables.   |
| <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="#">ServiceStart</a>  | Starts the specified service at the server to which Connection is connected.  |
| <a href="#">ServiceStatus</a>   | Returns a status of the specified service at the server to which Connection is connected.                               |
| <a href="#">ServiceStop</a>   | Stops the specified service at the server to which Connection is connected.   |
| <a href="#">SetOrderBy</a> (inherited from <a href="#">TCustomDADataset</a> )     | Builds an ORDER BY clause of a SELECT statement.  |
| <a href="#">ShowProcessList</a>   | Shows the list of the processes.  |
| <a href="#">ShowStatus</a>  | Shows the current state of the server.  |
| <a href="#">ShowVariables</a>   | Shows the values of some MySQL system variables that are in effect for the current connection.                          |
| <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="#">UnLockTable</a> (inherited from <a href="#">TCustomMyDataSet</a> )    | Releases a table locked by the <a href="#">TCustomMyDataSet.LockTable</a> method.  |
| <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**

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

© 1997-2012 Devart. All Rights Reserved.

Analyzes and stores the key distribution for the table

**Class**

[TMyServerControl](#)

**Syntax**

```
procedure AnalyzeTable;
```

**Remarks**

Call the AnalyzeTable method to analyze and store the key distribution for the table. During the analysis, the table is locked with a read lock. This works on MyISAM and BDB tables. This is equivalent to running *myisamchk -a* on the table. If the table hasn't changed since the last AnalyzeTable command, the table will not be analysed again. The list of the tables is used from [TableNames](#). Refer to the ANALYZE TABLE detailed description at MySQL Reference Manual.

**See Also**

- [TableNames](#)
- [OptimizeTable](#)
- [CheckTable](#)
- [RepairTable](#)

© 1997-2012 Devart. All Rights Reserved.

Checks MyISAM and InnoDB tables for failures.

## Class

[TMyServerControl](#)

## Syntax

```
procedure CheckTable (CheckTypes: TMyCheckTypes = [ctMedium]);
```

### Parameters

*CheckTypes*

Holds the type of the checking that will be performed.

## Remarks

Call the CheckTable method to check MyISAM and InnoDB tables for failures.

The list of the tables is used from [TableNames](#).

Refer to the CHECK TABLE detailed description at MySQL Reference Manual.

## See Also

- [TableNames](#)
- [AnalyzeTable](#)
- [OptimizeTable](#)
- [RepairTable](#)

© 1997-2012 Devart. All Rights Reserved.

Creates a new database with the specified name.

## Class

[TMyServerControl](#)

## Syntax

```
procedure CreateDatabase (DatabaseName: string; IfNotExists:
boolean = True; CharSetName: string = ''; CollationName: string
= '');
```

### Parameters

*DatabaseName*

Holds the name of the database that will be created.

*IfNotExists*

True, if a database with the specified name does not exist.

*CharsetName*

Holds the charset name.

*CollationName*

Holds the collation name.

## Remarks

Call the CreateDatabase method to create a new database with the specified name.

If IfNotExists is True, then a database will be created only if it did not exist.

CharsetName, CollationName parameters specify the language parameters of the created database.

## See Also

- [DropDatabase](#)

© 1997-2012 Devart. All Rights Reserved.

Drops the specified database.

## Class

[TMyServerControl](#)

## Syntax

```
procedure DropDatabase(DatabaseName: string; IfExists: boolean =  
True);
```

### Parameters

*DatabaseName*

Holds the database name.

*IfExists*

If a database with the specified name does not exist, an exception will be generated.

## Remarks

Call the DropDatabase method to drop the specified database.

IfExists parameter is used to raise an error message if the specified database doesn't exist.

**Note:** Use this function very carefully!

## See Also

- [DropDatabase](#)

© 1997-2012 Devart. All Rights Reserved.

Flushes server data from the memory to disk.

## Class

[TMyServerControl](#)

## Syntax

```
procedure Flush(FlushTypes: TMyFlushTypes);
```

### Parameters

*FlushTypes*

Holds the options that define the actions to take while performing flush operation.

## Remarks

Call the Flush method to flush forcibly server data from the memory to disk. Used if backup copy is needed.

Refer to FLUSH detailed description at MySQL Reference Manual.

© 1997-2012 Devart. All Rights Reserved.

Populates a list of MySQL services at the server to which Connection is connected.

## Class

[TMyServerControl](#)

## Syntax

```
procedure GetServiceNames(List: TStrings);
```

### Parameters

*List*

Holds a list of MySQL services.

### Remarks

Call the GetServiceNames property to populate a list of MySQL services at the server to which Connection is connected.

List.Objects[] fields are filled with the status of services in TMyServiceStatus format.

This method is available only for Windows.

**Note:** Any contents already in the target string list object are eliminated and overwritten by the data produced by GetServiceNames.

### See Also

- [ServiceStart](#)
  - [ServiceStop](#)
  - [ServiceStatus](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Breaks the execution of the specified process.

### Class

[TMyServerControl](#)

### Syntax

```
procedure KillProcess(ThreadId: integer);
```

#### Parameters

*ThreadId*

Holds the thread ID of the current connection.

### Remarks

Call the KillProcess method to break the execution of the specified process.

Refer to KILL detailed description at MySQL Reference Manual.

### See Also

- [ShowProcessList](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Packs tables deleting unused places from the file at the disk and defragmenting it.

### Class

[TMyServerControl](#)

### Syntax

```
procedure OptimizeTable;
```

### Remarks

Call the OptimizeTable method to pack tables deleting unused places from the file at the disk and defragmenting it.

It makes sense to call it from time to time when working with a table actively, especially when deleting frequently.

The list of the tables is used from [TableNames](#).

Refer to OPTIMIZE TABLE detailed description at MySQL Reference Manual.

### See Also

- [TableNames](#)



- [Analy eTable](#)
- [CheckTable](#)
- [RepairTable](#)

---

© 1997-2012 Devart. All Rights Reserved.

Repairs specified tables.

## Class

[TMyServerControl](#)

## Syntax

```
procedure RepairTable(RepairTypes: TMyRepairTypes = []);
```

### Parameters

*RepairTypes*

Holds the options that define the actions to take while performing repair operation on a table.

## Remarks

Call the RepairTable method to repair specified tables.

Works only with MyISAM tables.

List of the tables is used from [TableNames](#).

Refer to REPAIR TABLE detailed description at MySQL Reference Manual.

## See Also

- [TableNames](#)
- [Analy eTable](#)
- [Optimi eTable](#)
- [CheckTable](#)

---

© 1997-2012 Devart. All Rights Reserved.

Starts the specified service at the server to which Connection is connected.

## Class

[TMyServerControl](#)

## Syntax

```
procedure ServiceStart(const ServiceName: string; ParamStr: string  
= '');;
```

### Parameters

*ServiceName*

Holds the name of the service to start.

*ParamStr*

Holds the list of parameters with which service will be started.

## Remarks

Starts the specified service at the server to which Connection with ParamStr parameters is connected.

This method is available only for Windows.

## See Also

- [ServiceStop](#)
- [GetServiceNames](#)
- [ServiceStatus](#)

---

© 1997-2012 Devart. All Rights Reserved.

Returns a status of the specified service at the server to which Connection is connected.

### Class

[TMyServerControl](#)

### Syntax

```
function ServiceStatus(const ServiceName: string):
```

```
  TMyServiceStatus;
```

#### Parameters

*ServiceName*

Holds the name of the service to start.

#### Return Value

a status of the specified service.

### Remarks

Call the ServiceStatus method to return a status of the specified service at the server to which Connection is connected.

To get a list of MySQL services you should use [GetServiceNames](#) function.

This method is available only for Windows.

### See Also

- [ServiceStart](#)
  - [ServiceStop](#)
  - [GetServiceNames](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Stops the specified service at the server to which Connection is connected.

### Class

[TMyServerControl](#)

### Syntax

```
procedure ServiceStop(const ServiceName: string);
```

#### Parameters

*ServiceName*

Holds the name of the service to stop.

### Remarks

Call the ServiceStop method to stop the specified service at the server to which Connection is connected.

This method is available only for Windows.

### See Also

- [ServiceStart](#)
  - [GetServiceNames](#)
  - [ServiceStatus](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Shows the list of the processes.

### Class

[TMyServerControl](#)

### Syntax

```
procedure ShowProcessList(Full: boolean = False);
```

**Parameters***Full*

True, if full query text will be shown.

**Remarks**

Call the ShowProcessList method to show the list of the processes. Full parameter specifies whether to show the full text of the query or only the first 100 symbols.

To disconnect use the [KillProcess](#) method.

Refer to SHOW [FULL PROCESSLIST detailed description at MySQL Reference Manual.

**See Also**

- [KillProcess](#)

---

© 1997-2012 Devart. All Rights Reserved.

Shows the current state of the server.

**Class**

[TMyServerControl](#)

**Syntax**

```
procedure ShowStatus;
```

**Remarks**

Call the ShowStatus method to show the current state of the server.

**See Also**

- [ShowVariables](#)

---

© 1997-2012 Devart. All Rights Reserved.

Shows the values of some MySQL system variables that are in effect for the current connection.

**Class**

[TMyServerControl](#)

**Syntax**

```
procedure ShowVariables;
```

**Remarks**

Call the ShowVariables method to show the values of some MySQL system variables that are in effect for the current connection.

**See Also**

- [ShowStatus](#)

---

© 1997-2012 Devart. All Rights Reserved.

## **17.26 MySqlApi**

This unit contains implementation of the class.

### **Types**

| <b>Name</b>                 | <b>Description</b>   |
|-----------------------------|--|
| <a href="#">TMyLogEvent</a> | This type is used for the <a href="#">TMyEmbConnection.OnLog</a> and <a href="#">TMyEmbConnection.OnLogError</a> events. |

### **Variables**

| <b>Name</b>                        | <b>Description</b>  |
|------------------------------------|---|
| <a href="#">MySQLClientLibrary</a> | When set, specifies path and name of MySQL client library (libmysql.dll or libmysqld.dll for embedded server). Not exists for .NET Framework. |

---

## 17.26.1 Types

Types in the **MySqlApi** unit.

### Types

| Name                        | Description  |
|-----------------------------|--|
| <a href="#">TMyLogEvent</a> | This type is used for the <a href="#">TMyEmbConnection.OnLog</a> and <a href="#">TMyEmbConnection.OnLogError</a> events. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.26.1.1 MySqlApi.TMyLogEvent Procedure Reference

This type is used for the [TMyEmbConnection.OnLog](#) and [TMyEmbConnection.OnLogError](#) events.

### Unit

[MySqlApi](#)

### Syntax

```
TMyLogEvent = procedure (const Text: string) of object;
```

#### Parameters

*Text*

Holds the text of error, startup messages, entries that record client connections, and SQL statements.

© 1997-2012 Devart. All Rights Reserved.

## 17.26.2 Variables

Variables in the **MySqlApi** unit.

### Variables

| Name                               | Description   |
|------------------------------------|---|
| <a href="#">MySQLClientLibrary</a> | When set, specifies path and name of MySQL client library (libmysql.dll or libmysqld.dll for embedded server). Not exists for .NET Framework. |

---

© 1997-2012 Devart. All Rights Reserved.

#### 17.26.2.1 MySqlApi.MySQLClientLibrary Variable

When set, specifies path and name of MySQL client library (libmysql.dll or libmysqld.dll for embedded server). Not exists for .NET Framework.

### Unit

[MySqlApi](#)

### Syntax

```
MySQLClientLibrary: string;
```

---

© 1997-2012 Devart. All Rights Reserved.

## **17.27 MySQLMonitor**

This unit contains implementation of the TMySQLMonitor component.

### **Classes**

| <b>Name</b>                   | <b>Description</b>  |
|-------------------------------|---|
| <a href="#">TMySQLMonitor</a> | This component serves for monitoring dynamic SQL execution in MyDAC-based applications. |

## 17.27.1 Classes

Classes in the **MySQLMonitor** unit.

### Classes

| Name                          | Description   |
|-------------------------------|---|
| <a href="#">TMySQLMonitor</a> | This component serves for monitoring dynamic SQL execution in MyDAC-based applications. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.27.1.1 MySQLMonitor.TMySQLMonitor Class

This component serves for monitoring dynamic SQL execution in MyDAC-based applications. For a list of all members of this type, see [TMySQLMonitor](#) members.

### Unit

[MySQLMonitor](#)

### Syntax

```
TMySQLMonitor = class (TCustomMySQLMonitor) ;
```

### Remarks

Use TMySQLMonitor to monitor dynamic SQL execution in MyDAC-based applications. TMySQLMonitor provides two ways of displaying debug information: with dialog window, [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 TMySQLMonitor object. If an application has no TMySQLMonitor instance, the Debug window is available to display SQL statements to be sent.

### Inheritance Hierarchy

```

TObject
  TCustomDASQLMonitor
    TCustomMySQLMonitor
      TMySQLMonitor

```

© 1997-2012 Devart. All Rights Reserved.

[TMySQLMonitor](#) class overview.

© 1997-2012 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. |

## 17.28.1 Classes

Classes in the **VirtualTable** unit.

### Classes

| Name                          | Description                              |
|-------------------------------|--|
| <a href="#">TVirtualTable</a> | A base class for storing data in memory. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.28.1.1 VirtualTable.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 MySQL Access page.

### Inheritance Hierarchy

TObject

[TMemDataSet](#)

**TVirtualTable**

© 1997-2012 Devart. All Rights Reserved.

[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.               |

[Prepared](#) (inherited from [TMemDataSet](#))

[UpdateRecordTypes](#) (inherited from [TMemDataSet](#))

[UpdatesPending](#) (inherited from [TMemDataSet](#))

Determines whether a query is prepared for execution or not.

Used to indicate the update status for the current record when cached updates are enabled.

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.  |
| <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="#">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="#">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="#">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.                              |

[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.                     |

© 1997-2012 Devart. All Rights Reserved.

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 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.                           |

## 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](#)

© 1997-2012 Devart. All Rights Reserved.

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.

© 1997-2012 Devart. All Rights Reserved.

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.  |

[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.

[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

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

© 1997-2012 Devart. All Rights Reserved.

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](#)
- 

© 1997-2012 Devart. All Rights Reserved.

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

```
MyQuery1.SQL.Text := 'SELECT * FROM DEPT';  
MyQuery1.Active := True;  
VirtualTable1.Assign(MyQuery1);  
VirtualTable1.Active := True;
```

## See Also

- [TVirtualTable](#)
- 

© 1997-2012 Devart. All Rights Reserved.

Removes all records from TVirtualTable.

## Class

[TVirtualTable](#)

## Syntax

```
procedure Clear;
```

## Remarks

Call the Clear method to remove all records from TVirtualTable.

---

© 1997-2012 Devart. All Rights Reserved.

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](#)

---

© 1997-2012 Devart. All Rights Reserved.

Deletes all fields.

### Class

[TVirtualTable](#)

### Syntax

```
procedure DeleteFields;
```

### Remarks

Call the DeleteFields method to delete all fields.

### See Also

- [DeleteField](#)

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 Devart. All Rights Reserved.

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.

---

© 1997-2012 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> . |

---

© 1997-2012 Devart. All Rights Reserved.

#### 17.28.2.1 VirtualTable.TVirtualTableOptions Set

Represents the set of [TVirtualTableOption](#).

### Unit

[VirtualTable](#)

### Syntax

```
TVirtualTableOptions = set of TVirtualTableOption;
```

---

© 1997-2012 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. |

© 1997-2012 Devart. All Rights Reserved.

#### 17.28.3.1 VirtualTable.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. |

© 1997-2012 Devart. All Rights Reserved.

# Index

## - \_ -

\_\_Strings65535ToMemo Variable 543

## - 6 -

64-bit Development with Embarcadero RAD Studio XE2 81

## - A -

AbortOnKeyViol Property 95  
 AbortOnProblem Property 95  
 Active Property  
     TCustomDASQLMonitor 170  
     TDATransaction 301  
     TMacro 305  
 AddDBTypeRule Method 275  
 AddDrop Property 133  
 AddField Method 621  
 AddFieldNameRule Method 280  
 AddLock Property 559  
 AddRef Method 347  
 AddRule Method 281  
 AddWhere Method 226  
 AfterExecute Event  
     TCustomDADataSet 239  
     TCustomDASQL 253  
     TDAScript 159  
 AfterFetch Event 240  
 AfterUpdateExecute Event 240  
 AnalyzeTable Method 603  
 Apply Method 259  
 ApplyUpdates Method  
     ApplyUpdates 360, 361  
     TCustomDAConnection 195  
     TMemDataSet 360  
 AsBlob Property 291  
 AsBlobRef Property 292  
 AsDateTime Property 305  
 AsFloat Property  
     TDAParam 292  
     TMacro 305

AsInteger Property  
     TDAParam 292  
     TMacro 306  
 AsLargeInt Property 292  
 AsMemo Property 293  
 AsMemoRef Property 293  
 Assign Method  
     TBlob 336  
     TVirtualTable 622  
 AssignConnect Method 384  
 AssignField Method 295  
 AssignFieldValue Method 296  
 AssignValues Method 308  
 AsSQLTimeStamp Property 293  
 AsString Property  
     TBlob 334  
     TDAParam 293  
     TMacro 306  
 AsWideString Property  
     TBlob 335  
     TDAParam 294  
 AttributeByName Method 345  
 AttributeCount Property 343  
 AttributeNo Property 330  
 Attributes Property(Indexer) 343  
 AutoPrepare Property  
     TDADatasetOptions 265  
     TMyDataSetOptions 465  
 AutoRefresh Property 465  
 AutoRefreshInterval Property 466

## - B -

Backup Method  
     TDADump 128  
     TMyBackup 528  
 BackupPriority Property 523  
 BackupQuery Method 128  
 BackupToFile Method 128  
 BackupToStream Method 129  
 BaseDir Property 568  
 BaseSQL Property 212  
 BaseSQLOldBehavior Variable 319  
 bdError 533  
 bdIgnore 533  
 bdReplace 533  
 BeforeExecute Event 159  
 BeforeFetch Event 240

BeforeUpdateExecute Event 240  
 BinaryAsString Property 466  
 bmAppend 102  
 bmAppendUpdate 102  
 bmBinary 532  
 bmDelete 102  
 bmText 532  
 bmUpdate 102  
 bpConcurrent 532  
 bpDefault 532  
 bpLowPriority 532  
 BreakExec Method  
     TCustomDADataset 226  
     TDA Script 155  
     TMyCommand 448

## - C -

CACert Property 460  
 CacheCalcFields Property 265  
 CachedUpdates Property 356  
 CancelButton Property 184  
 CancelUpdates Method 361  
 Caption Property 184  
 Cert Property 460  
 ChangeCursor Property 243  
 ChangeCursor Variable 319  
 ChangedCount Property 96  
 Charset Property 390  
 CheckBackslashes Property 458  
 CheckRowVersion Property 466  
 CheckTable Method 604  
 ChipherList Property 461  
 clApply 350  
 clConnect 350  
 clConnectionApply 350  
 Clear Method  
     TBlob 337  
     TVirtualTable 622  
 clExecute 350  
 ClientVersion Property 379  
 clOpen 350  
 clRefresh 350  
 clServiceQuery 350  
 clTransStart 350  
 clUnknown 350  
 Columns Property 140  
 CommandTimeout Property  
     TCustomMyDataSet 401  
     TMyCommand 446  
 Commit Method  
     TCustomDAConnection 196  
     TDATransaction 302  
 CommitBatchSize Property 559  
 CommitCount Property 96  
 CommitUpdates Method 362  
 Compatibility 27  
 Compatibility with Previous Versions 74  
 Component List 23  
 Component Property 181  
 Compress Property 458  
 Connect Method 196  
 ConnectButton Property 185  
 ConnectDialog Property 190  
 Connection Pooling 62  
 Connection Property  
     TCustomDADataset 212  
     TCustomDASQL 243  
     TCustomMyDataSet 401  
     TDADump 125  
     TDALoader 140  
     TDAMetaData 286  
     TDA Script 151  
     TMyBackup 523  
     TMyBuilder 536  
     TMyCommand 446  
     TMyConnectDialog 549  
     TMyDump 555  
     TMyLoader 576  
     TMyScript 583  
     TMyServerControl 597  
 ConnectionLifetime Property 311  
 ConnectionTimeout Property 380  
 ConvertEOL Property 190  
 CRAccess Unit Members 88  
 CRBatchMove Unit Members 92  
 CRDataTypeMap Unit Members 103  
 CreateBlobStream Method 227  
 CreateColumns Method 141  
 CreateConnection Property 466  
 CreateDatabase Method 604  
 CreateDataSet Method  
     TCustomDAConnection 197  
     TCustomMyConnection 384  
 CreateSQL Method 197  
 CREncryption Unit Members 110

CRVio Unit Members 117

## - D -

DADDataAdapter Class 322

DADDataAdapter.DataSet Property 323

DADDataAdapter.Fill Method 323

DADDataAdapter.Update Method 324

DADump Unit Members 123

DALoader Unit Members 135

DAScript Unit Members 148

DASQLMonitor Unit Members 168

Data Encryption 58

Data Type Mapping 54

Database Property 380

Database Specific Aspects of 64-bit Development 85

DatabaseLabel Property 549

DataDir Property 569

DataHeader Property 112

DataSet Manager 66

DataSet Property

    DADDataAdapter 323

    TCustomDAUpdateSQL 255

    TDAScript 151

    TMyScript 584

DataSize Property 331

DataType Property

    TAttribute 331

    TDAParam 294

    TObjectType 344

DBAccess Unit Members 177

dbForge Fusion for MySQL 75

DBLengthMax Property

    TDAMapRule 272

    TMapRule 107

DBLengthMin Property

    TDAMapRule 272

    TMapRule 107

DBMonitor 71

DBMonitorOptions Property 170

DBScaleMax Property

    TDAMapRule 273

    TMapRule 107

DBScaleMin Property

    TDAMapRule 273

    TMapRule 108

DBType Property

TDAMapRule 273

TMapRule 108

Debug Property

    TCustomDADDataSet 212

    TCustomDASQL 244

TDADump 126

TDAScript 152

TMyBackup 524

TMyServerControl 597

DefaultCloseAction Property 301

DefaultSortType Property 261

DefaultValues Property

    TDADatasetOptions 265

    TMyDataSetOptions 467

DeferredPost Method 362

DeleteField Method 622

DeleteFields Method 623

DeleteObject Property 255

DeleteSQL Property 255

DeleteWhere Method 227

Delimiter Property 152

Demo Projects 18

Deployment 32

Destination Property 96

DetailDelay Property 265

DetailFields Property 213

Devart.Dac.DataAdapter Unit Members 321

Devart.MyDac.DataAdapter Unit Members 325

DialogClass Property 185

Direct Property 458

DisableKeys Property 559

Disconnect Method 197

Disconnected Mode 53

Disconnected Property 213

DisconnectedMode Property 261

dkIgnore 579

dkNone 579

dkReplace 579

doData 562

doDatabase 562

DoNotRaiseExcetionOnUaFail Variable 372

doStoredProcs 562

doTables 562

doTriggers 562

doUsers 562

doViews 562



DropDatabase Method 605  
 DuplicateKeys Property 577  
 Duplicates Property 524

## - E -

eaAbort 167  
 eaAES128 115  
 eaAES192 115  
 eaAES256 115  
 eaBlowfish 115  
 eaCast128 115  
 eaContinue 167  
 eaException 167  
 eaFail 167  
 eaRC4 115  
 eaTripleDES 115  
 EDAError Class 181  
 EDAError.Component Property 181  
 EDAError.ErrorCode Property 182  
 EDataMappingError Class 104  
 EDataTypeMappingError Class 104  
 Editions 3  
 ehNone 115  
 ehTag 115  
 ehTagAndHash 115  
 EInvalidDBTypeMapping Class 105  
 EInvalidFieldTypeMapping Class 105  
 Embedded Property 459  
 Embedded Server 49  
 EmptyTable Method 442  
 EMyError Class 540  
 EMyError.LineNumber Property 541  
 EnableBoolean Property 467  
 EnclosedBy Property 524  
 Encryption Property 213  
 EncryptionAlgorithm Property 112  
 Encryptor Property 270  
 EndLine Property  
     TDAScript 152  
     TDASStatement 161  
 EndOffset Property  
     TDAScript 152  
     TDASStatement 161  
 EndPos Property  
     TDAScript 153  
     TDASStatement 161  
 ErrorCode Property 182

ErrorOffset Method 156  
 EscapedBy Property 525  
 EUnsupportedDataTypeMapping Class 106  
 ExecProc Method  
     TCustomDAConnection 198  
     TCustomMyStoredProc 426  
 ExecProcEx Method 199  
 ExecSQL Method  
     TCustomDAConnection 199  
     TCustomDAUpdateSQL 259  
     TCustomMyConnection 385  
 ExecSQLEx Method 200  
 Execute Method  
     Execute 248, 249  
     TCRBatchMove 99  
     TCustomConnectDialog 187  
     TCustomDADataset 227  
     TCustomDASQL 248  
     TDAScript 156  
 ExecuteFile Method 156  
 ExecuteNext Method 157  
 ExecuteStream Method 157  
 Executing Method  
     TCustomDADataset 228  
     TCustomDASQL 249

## - F -

Features 9  
 FetchAll Property  
     TCustomMyDataSet 401  
     TMyQuery 484  
     TMyTable 506  
 Fetched Method 228  
 Fetching Method 229  
 FetchingAll Method 229  
 FetchRows Property 213  
 FieldLength Property  
     TDAMapRule 273  
     TMapRule 108  
 FieldMappingMode Property 96  
 FieldName Property  
     TDAMapRule 273  
     TMapRule 108  
 Fields Property  
     TDAEncryptionOptions 270  
     TMyBackup 525

- FieldsAsString Property 467
- FieldScale Property
  - TDAMapRule 274
  - TMapRule 108
- FieldsOrigin Property
  - TDADatasetOptions 266
  - TMyDatasetOptions 467
- FieldsTerminatedBy Property 526
- FieldType Property
  - TDAColumn 137
  - TDAMapRule 274
- Fill Method 323
- FilterSQL Property 214
- FinalSQL Property
  - TCustomDADataset 214
  - TCustomDASQL 244
- FindAttribute Method 345
- FindKey Method 229
- FindMacro Method
  - TCustomDADataset 230
  - TCustomDASQL 249
  - TDAScript 157
  - TMacros 309
- FindNearest Method 230
- FindParam Method
  - TCustomDADataset 231
  - TCustomDASQL 250
  - TDAParams 299
- FlatBuffers Property 266
- Flush Method 605
- Frequently Asked Questions 35
- FullRefresh Property 468

## - G -

- GenerateHeader Property 133
- GetBlob Method 362
- GetCharsetNames Method 385
- GetDatabaseNames Method 201
- GetDataType Method 231
- GetExecuteInfo Method 385
- GetFieldEnum Method 408
- GetFieldObject Method 231
- GetFieldPrecision Method 232
- GetFieldScale Method 232
- GetMetaDataKinds Method 288
- GetOrderBy Method 233
- GetRestrictions Method 289

- GetServerList Method 187
- GetServiceNames Method 605
- GetStoredProcNames Method 201
- Getting Started 5
- Getting Support 34
- GetTriggerNames Method 386
- GotoCurrent Method 233

## - H -

- haMD5 116
- HandlerIndex Property 512
- haSHA1 116
- HashAlgorithm Property 112
- HexBlob Property 559
- Hierarchy Chart 25
- Host Property 173
- Hostname Property 121
- HttpOptions Property 453

## - I -

- IgnoreErrors Property
  - TDAMapRule 274
  - TMapRule 108
- IgnoreLines Property 526
- ihFail 116
- ihIgnoreError 116
- ihSkipData 116
- ilReadCommitted 91
- ilReadUnCommitted 517
- ilRepeatableRead 517
- ilSerializable 517
- Increasing Performance 60
- IndexDefs Property 437
- IndexFieldNames Property 357
- InsertId Property
  - TCustomMyDataSet 402
  - TMyCommand 447
- InsertObject Property 256
- InsertSQL Property 256
- Installation 29
- Interactive Property 459
- InTransaction Property 190
- InvalidHashAction Property 113
- IOHandler Property 453
- IsEqual Method 309

IsNull Property 294  
IsolationLevel Property 380  
IsQuery Property 214  
IsUnicode Property 335  
Items Property(Indexer)  
    TDAColumns 138  
    TDAParams 299  
    TDASentences 164  
    TMacros 308

## - K -

KeepDesignConnected Property 261  
Key Property 461  
KeyFields Property 215  
KeyViolCount Property 97  
KillProcess Method 606

## - L -

LabelSet Property 185  
Length Property 331  
Licensing and Subscriptions 33  
Limit Property 437  
LineNumber Property 541  
LinesTerminatedBy Property 526  
ImNone 317  
ImOptimistic 317  
ImPessimistic 317  
Load Method 141  
LoadFromDataSet Method 142  
LoadFromFile Method  
    TBlob 337  
    TDAParam 296  
    TVirtualTable 623  
LoadFromStream Method  
    TBlob 337  
    TDAParam 297  
    TVirtualTable 624  
Local Property 527  
LocalConstraints Property 357  
LocalFailover Property 262  
LocalMasterDetail Property 266  
LocalUpdate Property 357  
Locate Method 363  
LocateEx Method 365  
Lock Method

Lock 409  
TCustomDADataset 233  
TCustomMyDataSet 409  
LockMode Property  
    TCustomMyDataSet 402  
TMyQuery 484  
TMyStoredProc 495  
TMyTable 507  
LockObject Property 256  
LockSQL Property 257  
LockTable Method 410  
IoDelayed 579  
LoginPrompt Property 191  
IoLock 579  
LongStrings Property 266  
IrDelayed 516  
IrImmediately 516  
IsCustom 317  
IsEnglish 317  
IsFrench 317  
IsGerman 317  
IsItalian 317  
IsPolish 317  
IsPortuguese 317  
IsRussian 317  
IsSpanish 317  
ItRead 516  
ItReadLocal 516  
ItWrite 516  
ItWriteLowPriority 516  
IxCaseInsensitive 351  
IxNearest 351  
IxNext 351  
IxPartialCompare 351  
IxPartialKey 351  
IxUp 351

## - M -

MacroByName Method  
    TCustomDADataset 234  
    TCustomDASQL 250  
    TDAScript 158  
    TMacros 309  
MacroChar Variable 320  
MacroCount Property  
    TCustomDADataset 215  
    TCustomDASQL 244

- Macros 64
- Macros Property
  - TCustomDADataSet 215
  - TCustomDASQL 245
  - TDAScript 153
- Mappings Property 97
- Master/Detail Relationships 41
- MasterFields Property 216
- MasterSource Property 216
- MaxPoolSize Property 311
- MemData Unit Members 328
- MemDS Unit Members 353
- MetaDataKind Property 286
- Migration from BDE 43
- Migration Wizard 72
- MinPoolSize Property 312
- mmFieldIndex 102
- mmFieldName 102
- moCustom 175
- moDBMonitor 175
- Mode Property
  - TCRBatchMove 97
  - TMyBackup 527
- moDialog 175
- ModifyObject Property 257
- ModifySQL Property 257
- moHandled 175
- MonitorMessage Method 202
- moSQLMonitor 175
- MovedCount Property 98
- mpDefault 542
- mpHttp 542
- mpMemory 542
- mpPipe 542
- mpSocket 542
- mpSSL 542
- mpTCP 542
- MyAccess Unit Members 373
- MyBackup Unit Members 520
- MyBuilder Add-In 80
- MyBuilderClient Unit Members 534
- MyClasses Unit Members 539
- MyConnectionPool Unit Members 544
- MyDacVcl Unit Members 546
- MydacVersion Constant 519
- MyDataAdapter Class 326
- MyDump Unit Members 552
- MyEmbConnection Unit Members 563

- MyLoader Unit Members 573
- MyScript Unit Members 580
- MyServerControl Unit Members 585
- MySqlApi Unit Members 610
- MySQLClientLibrary Variable 612
- MySQLMonitor Unit Members 613

## - N -

- Name Property
  - TDAColumn 137
  - TMacro 306
- National Characters 51
- Network Tunneling 47
- ntBCD 351
- ntFloat 351
- ntFmtBCD 351
- NullForZeroDate Property 468
- NullForZeroDelphiDate Property 390
- NumberRange Property
  - TDADatasetOptions 267
  - TMyDataSetOptions 468
- NumericType Property 390

## - O -

- Objects Property 556
- ObjectType Property 331
- Offset Property
  - TAttribute 332
  - TCustomMyTable 437
- Omit Property 162
- OnBackupProgress Event 131
- OnBatchMoveProgress Event 99
- OnConnectionLost Event 204
- OnError Event
  - TCustomDAConnection 204
  - TDADump 131
  - TDAScript 159
  - TDATransaction 303
- OnGetColumnData Event 144
- OnLog Event 572
- OnLogError Event 572
- OnProgress Event 144
- OnPutData Event 145
- OnRestoreProgress Event 132
- OnSQL Event 171

OnTableMsg Event 529  
 OnUpdateError Event 370  
 OnUpdateRecord Event 370  
 OptimizedBigInt Property 391  
 OptimizeTable Method 606  
 Options Property  
     TCustomDAConnection 191  
     TCustomDADataset 217  
     TCustomDASQLMonitor 170  
     TCustomMyConnection 381  
     TCustomMyDataSet 402  
     TCustomMyTable 437  
     TDADump 126  
     TMyConnection 454  
     TMyDump 556  
     TMyLoader 577  
     TVirtualTable 619  
 OrderFields Property 507  
 Overview 1  
 Owner Property 332

## - P -

ParamByName Method  
     TCustomDADataset 234  
     TCustomDASQL 251  
     TDAParams 300  
 ParamCheck Property  
     TCustomDADataset 218  
     TCustomDASQL 245  
 ParamCount Property  
     TCustomDADataset 219  
     TCustomDASQL 245  
 Params Property  
     TCustomDADataset 219  
     TCustomDASQL 246  
     TDASentence 162  
     TMyEmbConnection 569  
 ParamType Property 294  
 ParamValues Property(Indexer) 246  
 Password Property  
     TCREncryptor 113  
     TCustomDAConnection 192  
     THttpOptions 119  
     TProxyOptions 121  
 PasswordLabel Property 185  
 Path Property 527  
 Ping Method 387

Pooling Property 192  
 PoolingOptions Property 193  
 Port Property  
     TDBMonitorOptions 173  
     TMyConnection 455  
     TProxyOptions 121  
 PortLabel Property 550  
 Prepare Method  
     TCustomDADataset 235  
     TCustomDASQL 251  
     TMemDataSet 366  
 Prepared Property  
     TCustomDASQL 247  
     TMemDataSet 358  
 PrepareSQL Method 426  
 ProblemCount Property 98  
 Protocol Property 459  
 ProxyOptions Property 119  
 PutColumnData Method 142

## - Q -

QueryRecCount Property  
     TDADatasetOptions 267  
     TMyDataSetOptions 468  
 QuoteNames Property  
     TDADatasetOptions 267  
     TDADumpOptions 133  
     TMyDataSetOptions 469

## - R -

Read Method 338  
 ReadOnly Property 219  
 ReconnectTimeout Property 173  
 RecordCount Property 98  
 RefCount Property 347  
 RefreshObject Property 257  
 RefreshOptions Property 220  
 RefreshQuick Method 410  
 RefreshRecord Method 235  
 RefreshSQL Property 258  
 Release Method 348  
 ReleaseSavepoint Method 387  
 RemoveFromPool Method 202  
 RemoveOnRefresh Property  
     TDADatasetOptions 267

RemoveOnRefresh Property  
     TMyDataSetOptions 469  
 RepairTable Method 607  
 RequiredFields Property  
     TDADatasetOptions 268  
     TMyDataSetOptions 469  
 Requirements 26  
 Restore Method  
     TDADump 129  
     TMyBackup 529  
 RestoreFromFile Method 130  
 RestoreFromStream Method 130  
 RestoreSQL Method 236  
 RestoreUpdates Method 367  
 Restrictions Property 287  
 Resync Method 236  
 Retries Property 186  
 ReturnParams Property  
     TDADatasetOptions 268  
     TMyDataSetOptions 469  
 RevertRecord Method 367  
 rmRaise 318  
 rmReconnect 318  
 rmReconnectExecute 318  
 roAfterInsert 318  
 roAfterUpdate 318  
 roBeforeEdit 318  
 Rollback Method  
     TCustomDAConnection 202  
     TDATransaction 302  
 RollbackToSavepoint Method 387  
 RowsAffected Property  
     TCustomDADataset 220  
     TCustomDASQL 247  
 RowsPerQuery Property 577

## - S -

SavePassword Property 186  
 Savepoint Method 388  
 SaveSQL Method 236  
 SaveToFile Method  
     TBlob 338  
     TVirtualTable 624  
 SaveToStream Method  
     TBlob 338  
     TVirtualTable 624  
 SaveToXML Method 367

Scale Property 332  
 Scan Method 310  
 Script Property 162  
 Secure Connections 45  
 SendDataSetChangeEventAfterOpen  
 Variable 372  
 SendTimeout Property 173  
 Server Property 193  
 ServerLabel Property 186  
 ServerVersion Property 382  
 ServiceStart Method 607  
 ServiceStatus Method 608  
 ServiceStop Method 608  
 SetBlobData 297  
 SetBlobData Method 297  
 SetFieldsReadOnly Property  
     TDADatasetOptions 268  
     TMyDataSetOptions 470  
 SetKey Method 113  
 SetOrderBy Method 237  
 Show Method 537  
 ShowDatabase Property 550  
 ShowModal Method 538  
 ShowPort Property 550  
 ShowProcessList Method 608  
 ShowStatus Method 609  
 ShowVariables Method 609  
 Size Property  
     TAttribute 332  
     TBlob 335  
     TDAParam 295  
     TObjectType 344  
 Source Property 99  
 SQL Property  
     TCustomDADataset 220  
     TCustomDASQL 247  
     TDADump 127  
     TDAScript 153  
     TDASTatement 162  
     TMyBuilder 536  
 SQL Property(Indexer) 258  
 SQLDelete Property 221  
 SQLGeneratorCompatibility Variable 320  
 SQLInsert Property 221  
 SQLLock Property 221  
 SQLRefresh Property 222  
 SQLSaved Method 237  
 SQLUpdate Property 223

- SSLOptions Property 455
- StartLine Property
  - TDAScript 153
  - TDASStatement 163
- StartOffset Property
  - TDAScript 154
  - TDASStatement 163
- StartPos Property
  - TDAScript 154
  - TDASStatement 163
- StartTransaction Method
  - TCustomDAConnection 203
  - TDATransaction 303
- Statements Property 154
- stBinary 351
- stCaseInsensitive 351
- stCaseSensitive 351
- StoredProcName Property 421
- StoredProcNames Property 557
- StoreLogInfo Property 186
- StrictUpdate Property
  - TDADatasetOptions 269
  - TMyDataSetOptions 470

## - T -

- TableName Property
  - TDALoader 140
  - TMyTable 507
- TableNames Property
  - TDADump 127
  - TMyBackup 528
  - TMyServerControl 597
- taCommit 91
- TAfterExecuteEvent Procedure Reference 313
- TAfterFetchEvent Procedure Reference 313
- TAfterStatementExecuteEvent Procedure Reference 165
- taRollback 91
- TAttribute Class 329
- TAttribute.AttributeNo Property 330
- TAttribute.DataSize Property 331
- TAttribute.DataType Property 331
- TAttribute.Length Property 331
- TAttribute.ObjectType Property 331
- TAttribute.Offset Property 332
- TAttribute.Owner Property 332
- TAttribute.Scale Property 332
- TAttribute.Size Property 332
- TBeforeFetchEvent Procedure Reference 314
- TBeforeFetchProc Procedure Reference 90
- TBeforeStatementExecuteEvent Procedure Reference 165
- TBlob Class 333
- TBlob.Assign Method 336
- TBlob.AsString Property 334
- TBlob.AsWideString Property 335
- TBlob.Clear Method 337
- TBlob.IsUnicode Property 335
- TBlob.LoadFromFile Method 337
- TBlob.LoadFromStream Method 337
- TBlob.Read Method 338
- TBlob.SaveToFile Method 338
- TBlob.SaveToStream Method 338
- TBlob.Size Property 335
- TBlob.Truncate Method 339
- TBlob.Write Method 339
- TCompressedBlob Class 340
- TConnectionLostEvent Procedure Reference 314
- TConnLostCause Enumeration 350
- TCRBatchMode Enumeration 102
- TCRBatchMove Class 93
- TCRBatchMove.AbortOnKeyViol Property 95
- TCRBatchMove.AbortOnProblem Property 95
- TCRBatchMove.ChangedCount Property 96
- TCRBatchMove.CommitCount Property 96
- TCRBatchMove.Destination Property 96
- TCRBatchMove.Execute Method 99
- TCRBatchMove.FieldMappingMode Property 96
- TCRBatchMove.KeyViolCount Property 97
- TCRBatchMove.Mappings Property 97
- TCRBatchMove.Mode Property 97
- TCRBatchMove.MovedCount Property 98
- TCRBatchMove.OnBatchMoveProgress Event 99
- TCRBatchMove.ProblemCount Property 98
- TCRBatchMove.RecordCount Property 98
- TCRBatchMove.Source Property 99



- TCRBatchMoveProgressEvent Procedure Reference 101
- TCRCursor Class 89
- TCRDataSource Class 182
- TCREncDataHeader Enumeration 115
- TCREncryptionAlgorithm Enumeration 115
- TCREncryptor Class 111
- TCREncryptor.DataHeader Property 112
- TCREncryptor.EncryptionAlgorithm Property 112
- TCREncryptor.HashAlgorithm Property 112
- TCREncryptor.InvalidHashAction Property 113
- TCREncryptor.Password Property 113
- TCREncryptor.SetKey Method 113
- TCRFieldMappingMode Enumeration 102
- TCRHashAlgorithm Enumeration 116
- TCRInvalidHashAction Enumeration 116
- TCRIsoLevel Enumeration 91
- TCRTransactionAction Enumeration 91
- TCustomConnectDialog Class 182
- TCustomConnectDialog.CancelButton Property 184
- TCustomConnectDialog.Caption Property 184
- TCustomConnectDialog.ConnectButton Property 185
- TCustomConnectDialog.DialogClass Property 185
- TCustomConnectDialog.Execute Method 187
- TCustomConnectDialog.GetServerList Method 187
- TCustomConnectDialog.LabelSet Property 185
- TCustomConnectDialog.PasswordLabel Property 185
- TCustomConnectDialog.Retries Property 186
- TCustomConnectDialog.SavePassword Property 186
- TCustomConnectDialog.ServerLabel Property 186
- TCustomConnectDialog.StoreLogInfo Property 186
- TCustomConnectDialog.UsernameLabel Property 187
- TCustomDAConnection Class 188
- TCustomDAConnection.ApplyUpdates Method 195
- TCustomDAConnection.Commit Method 196
- TCustomDAConnection.Connect Method 196
- TCustomDAConnection.ConnectDialog Property 190
- TCustomDAConnection.ConvertEOL Property 190
- TCustomDAConnection.CreateDataSet Method 197
- TCustomDAConnection.CreateSQL Method 197
- TCustomDAConnection.Disconnect Method 197
- TCustomDAConnection.ExecProc Method 198
- TCustomDAConnection.ExecProcEx Method 199
- TCustomDAConnection.ExecSQL Method 199
- TCustomDAConnection.ExecSQLEx Method 200
- TCustomDAConnection.GetDatabaseNames Method 201
- TCustomDAConnection.GetStoredProcNames Method 201
- TCustomDAConnection.InTransaction Property 190
- TCustomDAConnection.LoginPrompt Property 191
- TCustomDAConnection.MonitorMessage Method 202
- TCustomDAConnection.OnConnectionLost Event 204
- TCustomDAConnection.OnError Event 204
- TCustomDAConnection.Options Property 191
- TCustomDAConnection.Password Property 192
- TCustomDAConnection.Pooling Property 192
- TCustomDAConnection.PoolingOptions Property 193
- TCustomDAConnection.RemoveFromPool Method 202
- TCustomDAConnection.Rollback Method 202
- TCustomDAConnection.Server Property 193



- TCustomDAConnection.StartTransaction Method 203
- TCustomDAConnection.Username Property 194
- TCustomDADataset Class 204
- TCustomDADataset.AddWhere Method 226
- TCustomDADataset.AfterExecute Event 239
- TCustomDADataset.AfterFetch Event 240
- TCustomDADataset.AfterUpdateExecute Event 240
- TCustomDADataset.BaseSQL Property 212
- TCustomDADataset.BeforeFetch Event 240
- TCustomDADataset.BeforeUpdateExecute Event 240
- TCustomDADataset.BreakExec Method 226
- TCustomDADataset.Connection Property 212
- TCustomDADataset.CreateBlobStream Method 227
- TCustomDADataset.Debug Property 212
- TCustomDADataset.DeleteWhere Method 227
- TCustomDADataset.DetailFields Property 213
- TCustomDADataset.Disconnected Property 213
- TCustomDADataset.Encryption Property 213
- TCustomDADataset.Execute Method 227
- TCustomDADataset.Executing Method 228
- TCustomDADataset.Fetched Method 228
- TCustomDADataset.Fetching Method 229
- TCustomDADataset.FetchingAll Method 229
- TCustomDADataset.FetchRows Property 213
- TCustomDADataset.FilterSQL Property 214
- TCustomDADataset.FinalSQL Property 214
- TCustomDADataset.FindKey Method 229
- TCustomDADataset.FindMacro Method 230
- TCustomDADataset.FindNearest Method 230
- TCustomDADataset.FindParam Method 231
- TCustomDADataset.GetDataType Method 231
- TCustomDADataset.GetFieldObject Method 231
- TCustomDADataset.GetFieldPrecision Method 232
- TCustomDADataset.GetFieldScale Method 232
- TCustomDADataset.GetOrderBy Method 233
- TCustomDADataset.GotoCurrent Method 233
- TCustomDADataset.IsQuery Property 214
- TCustomDADataset.KeyFields Property 215
- TCustomDADataset.Lock Method 233
- TCustomDADataset.MacroByName Method 234
- TCustomDADataset.MacroCount Property 215
- TCustomDADataset.Macros Property 215
- TCustomDADataset.MasterFields Property 216
- TCustomDADataset.MasterSource Property 216
- TCustomDADataset.Options Property 217
- TCustomDADataset.ParamByName Method 234
- TCustomDADataset.ParamCheck Property 218
- TCustomDADataset.ParamCount Property 219
- TCustomDADataset.Params Property 219
- TCustomDADataset.Prepare Method 235
- TCustomDADataset.ReadOnly Property 219
- TCustomDADataset.RefreshOptions Property 220
- TCustomDADataset.RefreshRecord Method 235
- TCustomDADataset.RestoreSQL Method 236
- TCustomDADataset.Resync Method 236
- TCustomDADataset.RowsAffected Property 220
- TCustomDADataset.SaveSQL Method 236
- TCustomDADataset.SetOrderBy Method 237
- TCustomDADataset.SQL Property 220

- TCustomDADataSet.SQLDelete Property 221
- TCustomDADataSet.SQLInsert Property 221
- TCustomDADataSet.SQLLock Property 221
- TCustomDADataSet.SQLRefresh Property 222
- TCustomDADataSet.SQLSaved Method 237
- TCustomDADataSet.SQLUpdate Property 223
- TCustomDADataSet.UniDirectional Property 223
- TCustomDADataSet.Unlock Method 237
- TCustomDASQL Class 241
- TCustomDASQL.AfterExecute Event 253
- TCustomDASQL.ChangeCursor Property 243
- TCustomDASQL.Connection Property 243
- TCustomDASQL.Debug Property 244
- TCustomDASQL.Execute Method 248
- TCustomDASQL.Executing Method 249
- TCustomDASQL.FinalSQL Property 244
- TCustomDASQL.FindMacro Method 249
- TCustomDASQL.FindParam Method 250
- TCustomDASQL.MacroByName Method 250
- TCustomDASQL.MacroCount Property 244
- TCustomDASQL.Macros Property 245
- TCustomDASQL.ParamByName Method 251
- TCustomDASQL.ParamCheck Property 245
- TCustomDASQL.ParamCount Property 245
- TCustomDASQL.Params Property 246
- TCustomDASQL.ParamValues Property(Indexer) 246
- TCustomDASQL.Prepare Method 251
- TCustomDASQL.Prepared Property 247
- TCustomDASQL.RowsAffected Property 247
- TCustomDASQL.SQL Property 247
- TCustomDASQL.UnPrepare Method 252
- TCustomDASQL.WaitExecuting Method 252
- TCustomDASQLMonitor Class 169
- TCustomDASQLMonitor.Active Property 170
- TCustomDASQLMonitor.DBMonitorOptions Property 170
- TCustomDASQLMonitor.OnSQL Event 171
- TCustomDASQLMonitor.Options Property 170
- TCustomDASQLMonitor.TraceFlags Property 171
- TCustomDAUpdateSQL Class 253
- TCustomDAUpdateSQL.Apply Method 259
- TCustomDAUpdateSQL.DataSet Property 255
- TCustomDAUpdateSQL.DeleteObject Property 255
- TCustomDAUpdateSQL.DeleteSQL Property 255
- TCustomDAUpdateSQL.ExecSQL Method 259
- TCustomDAUpdateSQL.InsertObject Property 256
- TCustomDAUpdateSQL.InsertSQL Property 256
- TCustomDAUpdateSQL.LockObject Property 256
- TCustomDAUpdateSQL.LockSQL Property 257
- TCustomDAUpdateSQL.ModifyObject Property 257
- TCustomDAUpdateSQL.ModifySQL Property 257
- TCustomDAUpdateSQL.RefreshObject Property 257
- TCustomDAUpdateSQL.RefreshSQL Property 258
- TCustomDAUpdateSQL.SQL Property(Indexer) 258
- TCustomMyConnection Class 376
- TCustomMyConnection.AssignConnect Method 384
- TCustomMyConnection.ClientVersion Property 379
- TCustomMyConnection.ConnectionTimeout Property 380
- TCustomMyConnection.CreateDataSet Method 384
- TCustomMyConnection.Database Property 380
- TCustomMyConnection.ExecSQL Method 385

- TCustomMyConnection.GetCharsetNames Method 385
- TCustomMyConnection.GetExecutefInfo Method 385
- TCustomMyConnection.GetTriggerNames Method 386
- TCustomMyConnection.IsolationLevel Property 380
- TCustomMyConnection.Options Property 381
- TCustomMyConnection.Ping Method 387
- TCustomMyConnection.ReleaseSavepoint Method 387
  
- TCustomMyConnection.RollbackToSavepoint Method 387
- TCustomMyConnection.Savepoint Method 388
- TCustomMyConnection.ServerVersion Property 382
- TCustomMyConnection.ThreadId Property 382
- TCustomMyConnectionOptions Class 388
- TCustomMyConnectionOptions.Charset Property 390
  
- TCustomMyConnectionOptions.NullForZero DelphiDate Property 390
  
- TCustomMyConnectionOptions.NumericType Property 390
  
- TCustomMyConnectionOptions.OptimizedBigInt Property 391
  
- TCustomMyConnectionOptions.UseUnicode Property 391
- TCustomMyDataSet Class 391
- TCustomMyDataSet.CommandTimeout Property 401
- TCustomMyDataSet.Connection Property 401
- TCustomMyDataSet.FetchAll Property 401
- TCustomMyDataSet.GetFieldEnum Method 408
- TCustomMyDataSet.InsertId Property 402
- TCustomMyDataSet.Lock Method 409
- TCustomMyDataSet.LockMode Property 402
- TCustomMyDataSet.LockTable Method 410
- TCustomMyDataSet.Options Property 402
  
- TCustomMyDataSet.RefreshQuick Method 410
- TCustomMyDataSet.UnlockTable Method 411
- TCustomMyStoredProc Class 411
- TCustomMyStoredProc.ExecProc Method 426
- TCustomMyStoredProc.PrepareSQL Method 426
- TCustomMyStoredProc.StoredProcName Property 421
- TCustomMyTable Class 427
- TCustomMyTable.EmptyTable Method 442
- TCustomMyTable.IndexDefs Property 437
- TCustomMyTable.Limit Property 437
- TCustomMyTable.Offset Property 437
- TCustomMyTable.Options Property 437
- TDABackupProgressEvent Procedure Reference 134
- TDAColumn Class 136
- TDAColumn.FieldType Property 137
- TDAColumn.Name Property 137
- TDAColumns Class 137
- TDAColumns.Items Property(Indexer) 138
- TDAConnectionErrorEvent Procedure Reference 314
- TDAConnectionOptions Class 260
- TDAConnectionOptions.DefaultSortType Property 261
  
- TDAConnectionOptions.DisconnectedMode Property 261
  
- TDAConnectionOptions.KeepDesignConnected Property 261
- TDAConnectionOptions.LocalFailover Property 262
- TDADatasetOptions Class 262
- TDADatasetOptions.AutoPrepare Property 265
- TDADatasetOptions.CacheCalcFields Property 265
- TDADatasetOptions.DefaultValues Property 265
- TDADatasetOptions.DetailDelay Property 265
- TDADatasetOptions.FieldsOrigin Property 266
- TDADatasetOptions.FlatBuffers Property 266

- TDADatasetOptions.LocalMasterDetail Property 266
- TDADatasetOptions.LongStrings Property 266
- TDADatasetOptions.NumberRange Property 267
- TDADatasetOptions.QueryRecCount Property 267
- TDADatasetOptions.QuoteNames Property 267
- TDADatasetOptions.RemoveOnRefresh Property 267
- TDADatasetOptions.RequiredFields Property 268
- TDADatasetOptions.ReturnParams Property 268
- TDADatasetOptions.SetFieldsReadOnly Property 268
- TDADatasetOptions.StrictUpdate Property 269
- TDADatasetOptions.UpdateAllFields Property 269
- TDADatasetOptions.UpdateBatchSize Property 269
- TDADump Class 124
- TDADump.Backup Method 128
- TDADump.BackupQuery Method 128
- TDADump.BackupToFile Method 128
- TDADump.BackupToStream Method 129
- TDADump.Connection Property 125
- TDADump.Debug Property 126
- TDADump.OnBackupProgress Event 131
- TDADump.OnError Event 131
- TDADump.OnRestoreProgress Event 132
- TDADump.Options Property 126
- TDADump.Restore Method 129
- TDADump.RestoreFromFile Method 130
- TDADump.RestoreFromStream Method 130
- TDADump.SQL Property 127
- TDADump.TableNames Property 127
- TDADumpOptions Class 132
- TDADumpOptions.AddDrop Property 133
- TDADumpOptions.GenerateHeader Property 133
- TDADumpOptions.QuoteNames Property 133
- TDAEncryptionOptions Class 269
- TDAEncryptionOptions.Encryptor Property 270
- TDAEncryptionOptions.Fields Property 270
- TDALoader Class 138
- TDALoader.Columns Property 140
- TDALoader.Connection Property 140
- TDALoader.CreateColumns Method 141
- TDALoader.Load Method 141
- TDALoader.LoadFromDataSet Method 142
- TDALoader.OnGetColumnData Event 144
- TDALoader.OnProgress Event 144
- TDALoader.OnPutData Event 145
- TDALoader.PutColumnData Method 142
- TDALoader.TableName Property 140
- TDAMapRule Class 271
- TDAMapRule.DBLengthMax Property 272
- TDAMapRule.DBLengthMin Property 272
- TDAMapRule.DBScaleMax Property 273
- TDAMapRule.DBScaleMin Property 273
- TDAMapRule.DBType Property 273
- TDAMapRule.FieldLength Property 273
- TDAMapRule.FieldName Property 273
- TDAMapRule.FieldScale Property 274
- TDAMapRule.FieldType Property 274
- TDAMapRule.IgnoreErrors Property 274
- TDAMapRules Class 274
- TDAMapRules.AddDBTypeRule Method 276
- TDAMapRules.AddFieldNameRule Method 280
- TDAMapRules.AddRule Method 281
- TDAMetaData Class 282
- TDAMetaData.Connection Property 286
- TDAMetaData.GetMetaDataKinds Method 288
- TDAMetaData.GetRestrictions Method 289
- TDAMetaData.MetaDataKind Property 286
- TDAMetaData.Restrictions Property 287
- TDANumericType Enumeration 351
- TDAParam Class 289
- TDAParam.AsBlob Property 291
- TDAParam.AsBlobRef Property 292
- TDAParam.AsFloat Property 292
- TDAParam.AsInteger Property 292
- TDAParam.AsLargeInt Property 292
- TDAParam.AsMemo Property 293
- TDAParam.AsMemoRef Property 293
- TDAParam.AssignField Method 295

TDAParam.AssignFieldValue Method 296  
TDAParam.AsSQLTimeStamp Property 293  
TDAParam.AsString Property 293  
TDAParam.AsWideString Property 294  
TDAParam.DataType Property 294  
TDAParam.IsNull Property 294  
TDAParam.LoadFromFile Method 296  
TDAParam.LoadFromStream Method 297  
TDAParam.ParamType Property 294  
TDAParam.SetBlobData 297  
TDAParam.SetBlobData Method 297  
TDAParam.Size Property 295  
TDAParam.Value Property 295  
TDAParams Class 298  
TDAParams.FindParam Method 299  
TDAParams.Items Property(Indexer) 299  
TDAParams.ParamByName Method 300  
TDAPutDataEvent Procedure Reference 146  
TDARestoreProgressEvent Procedure Reference 134  
TDA Script Class 149  
TDA Script.AfterExecute Event 159  
TDA Script.BeforeExecute Event 159  
TDA Script.BreakExec Method 155  
TDA Script.Connection Property 151  
TDA Script.DataSet Property 151  
TDA Script.Debug Property 152  
TDA Script.Delimiter Property 152  
TDA Script.EndLine Property 152  
TDA Script.EndOffset Property 152  
TDA Script.EndPos Property 153  
TDA Script.ErrorOffset Method 156  
TDA Script.Execute Method 156  
TDA Script.ExecuteFile Method 156  
TDA Script.ExecuteNext Method 157  
TDA Script.ExecuteStream Method 157  
TDA Script.FindMacro Method 157  
TDA Script.MacroByName Method 158  
TDA Script.Macros Property 153  
TDA Script.OnError Event 159  
TDA Script.SQL Property 153  
TDA Script.StartLine Property 153  
TDA Script.StartOffset Property 154  
TDA Script.StartPos Property 154  
TDA Script.Statements Property 154  
TDAStatement Class 159  
TDAStatement.EndLine Property 161  
TDAStatement.EndPos Property 161  
TDAStatement.Omit Property 162  
TDAStatement.Params Property 162  
TDAStatement.Script Property 162  
TDAStatement.SQL Property 162  
TDAStatement.StartLine Property 163  
TDAStatement.StartOffset Property 163  
TDAStatement.StartPos Property 163  
TDAStatements Class 163  
TDAStatements.Items Property(Indexer) 164  
TDATraceFlag Enumeration 175  
TDATraceFlags Set 174  
TDATransaction Class 300  
TDATransaction.Active Property 301  
TDATransaction.Commit Method 302  
TDATransaction.DefaultCloseAction Property 301  
TDATransaction.OnError Event 303  
TDATransaction.Rollback Method 302  
TDATransaction.StartTransaction Method 303  
TDATransactionErrorEvent Procedure Reference 315  
TDBMonitorOptions Class 172  
TDBMonitorOptions.Host Property 173  
TDBMonitorOptions.Port Property 173  
TDBMonitorOptions.ReconnectTimeout Property 173  
TDBMonitorOptions.SendTimeout Property 173  
TDBObject Class 341  
TErrorAction Enumeration 167  
tfBlob 175  
tfConnect 175  
tfError 175  
tfMisc 175  
tfObjDestroy 175  
tfParams 175  
tfPool 175  
tfQExecute 175  
tfQFetch 175  
tfQPrepare 175  
tfService 175  
tfStmt 175  
tfTransact 175  
TGetColumnDataEvent Procedure Reference 146



- ThreadId Property 382
- THttpOptions Class 118
- THttpOptions.Password Property 119
- THttpOptions.ProxyOptions Property 119
- THttpOptions.Url Property 119
- THttpOptions.Username Property 120
- TLabelSet Enumeration 317
- TLoaderProgressEvent Procedure Reference 147
- TLocateExOption Enumeration 351
- TLocateExOptions Set 349
- TLockMode Enumeration 317
- TLockRecordType Enumeration 516
- TLockType Enumeration 516
- TMacro Class 304
- TMacro.Active Property 305
- TMacro.AsDateTime Property 305
- TMacro.AsFloat Property 305
- TMacro.AsInteger Property 306
- TMacro.AsString Property 306
- TMacro.Name Property 306
- TMacro.Value Property 306
- TMacros Class 307
- TMacros.AssignValues Method 308
- TMacros.FindMacro Method 309
- TMacros.IsEqual Method 309
- TMacros.Items Property(Indexer) 308
- TMacros.MacroByName Method 309
- TMacros.Scan Method 310
- TMapRule Class 106
- TMapRule.DBLengthMax Property 107
- TMapRule.DBLengthMin Property 107
- TMapRule.DBScaleMax Property 107
- TMapRule.DBScaleMin Property 108
- TMapRule.DBType Property 108
- TMapRule.FieldLength Property 108
- TMapRule.FieldName Property 108
- TMapRule.FieldScale Property 108
- TMapRule.IgnoreErrors Property 108
- TMemDataSet Class 354
- TMemDataSet.ApplyUpdates Method 360
- TMemDataSet.CachedUpdates Property 356
- TMemDataSet.CancelUpdates Method 361
- TMemDataSet.CommitUpdates Method 362
- TMemDataSet.DeferredPost Method 362
- TMemDataSet.GetBlob Method 363
- TMemDataSet.IndexFieldNames Property 357
- TMemDataSet.LocalConstraints Property 357
- TMemDataSet.LocalUpdate Property 357
- TMemDataSet.Locate Method 364
- TMemDataSet.LocateEx Method 365
- TMemDataSet.OnUpdateError Event 370
- TMemDataSet.OnUpdateRecord Event 370
- TMemDataSet.Prepare Method 366
- TMemDataSet.Prepared Property 358
- TMemDataSet.RestoreUpdates Method 367
- TMemDataSet.RevertRecord Method 367
- TMemDataSet.SaveToXML Method 368
- TMemDataSet.UnPrepare Method 368
- TMemDataSet.UpdateRecordTypes Property 358
- TMemDataSet.UpdateResult Method 369
- TMemDataSet.UpdatesPending Property 358
- TMemDataSet.UpdateStatus Method 369
- TMonitorOption Enumeration 175
- TMonitorOptions Set 174
- TMyBackup Class 521
- TMyBackup.Backup Method 528
- TMyBackup.BackupPriority Property 523
- TMyBackup.Connection Property 523
- TMyBackup.Debug Property 524
- TMyBackup.Duplicates Property 524
- TMyBackup.EnclosedBy Property 524
- TMyBackup.EscapedBy Property 525
- TMyBackup.Fields Property 525
- TMyBackup.FieldsTerminatedBy Property 526
- TMyBackup.IgnoreLines Property 526
- TMyBackup.LinesTerminatedBy Property 526
- TMyBackup.Local Property 527
- TMyBackup.Mode Property 527
- TMyBackup.OnTableMsg Event 529
- TMyBackup.Path Property 527
- TMyBackup.Restore Method 529
- TMyBackup.TableNames Property 528
- TMyBackupMode Enumeration 532
- TMyBackupPriority Enumeration 532
- TMyBuilder Class 535
- TMyBuilder.Connection Property 536
- TMyBuilder.Show Method 537

- TMyBuilder.ShowModal Method 538
- TMyBuilder.SQL Property 536
- TMyBuilder.Version Property 537
- TMyColumn Class 574
- TMyCommand Class 443
- TMyCommand.BreakExec Method 448
- TMyCommand.CommandTimeout Property 446
- TMyCommand.Connection Property 446
- TMyCommand.InsertId Property 447
- TMyConnectDialog Class 547
- TMyConnectDialog.Connection Property 549
- TMyConnectDialog.DatabaseLabel Property 549
- TMyConnectDialog.PortLabel Property 550
- TMyConnectDialog.ShowDatabase Property 550
- TMyConnectDialog.ShowPort Property 550
- TMyConnection Class 449
- TMyConnection.HttpOptions Property 453
- TMyConnection.IOHandler Property 453
- TMyConnection.Options Property 454
- TMyConnection.Port Property 455
- TMyConnection.SSLOptions Property 455
- TMyConnectionOptions Class 455
- TMyConnectionOptions.CheckBackslashes Property 458
- TMyConnectionOptions.Compress Property 458
- TMyConnectionOptions.Direct Property 458
- TMyConnectionOptions.Embedded Property 459
- TMyConnectionOptions.Interactive Property 459
- TMyConnectionOptions.Protocol Property 459
- TMyConnectionPoolManager Class 545
- TMyConnectionSSLOptions Class 459
- TMyConnectionSSLOptions.CACert Property 460
- TMyConnectionSSLOptions.Cert Property 460
- TMyConnectionSSLOptions.ChipherList Property 461
- TMyConnectionSSLOptions.Key Property 461
- TMyDataSetOptions Class 461
- TMyDataSetOptions.AutoPrepare Property 465
- TMyDataSetOptions.AutoRefresh Property 465
- TMyDataSetOptions.AutoRefreshInterval Property 466
- TMyDataSetOptions.BinaryAsString Property 466
- TMyDataSetOptions.CheckRowVersion Property 466
- TMyDataSetOptions.CreateConnection Property 466
- TMyDataSetOptions.DefaultValues Property 467
- TMyDataSetOptions.EnableBoolean Property 467
- TMyDataSetOptions.FieldsAsString Property 467
- TMyDataSetOptions.FieldsOrigin Property 467
- TMyDataSetOptions.FullRefresh Property 468
- TMyDataSetOptions.NullForZeroDate Property 468
- TMyDataSetOptions.NumberRange Property 468
- TMyDataSetOptions.QueryRecCount Property 468
- TMyDataSetOptions.QuoteNames Property 469
- TMyDataSetOptions.RemoveOnRefresh Property 469
- TMyDataSetOptions.RequiredFields Property 469
- TMyDataSetOptions.ReturnParams Property 469
- TMyDataSetOptions.SetFieldsReadOnly Property 470
- TMyDataSetOptions.StrictUpdate Property 470
- TMyDataSetOptions.TrimFixedChar Property 470
- TMyDataSource Class 470
- TMyDump Class 553
- TMyDump.Connection Property 555
- TMyDump.Objects Property 556
- TMyDump.Options Property 556
- TMyDump.StoredProcNames Property 557
- TMyDump.TriggerNames Property 557
- TMyDumpObject Enumeration 562
- TMyDumpObjects Set 561

- TMyDumpOptions Class 557
- TMyDumpOptions.AddLock Property 559
- TMyDumpOptions.CommitBatchSize Property 559
- TMyDumpOptions.DisableKeys Property 559
- TMyDumpOptions.HexBlob Property 559
- TMyDumpOptions.UseDelayedIns Property 560
- TMyDumpOptions.UseExtSyntax Property 560
- TMyDuplicateKeys Enumeration 579
- TMyEmbConnection Class 564
- TMyEmbConnection.BaseDir Property 568
- TMyEmbConnection.DataDir Property 569
- TMyEmbConnection.OnLog Event 572
- TMyEmbConnection.OnLogError Event 572
- TMyEmbConnection.Params Property 569
- TMyEncryptor Class 471
- TMyIsolationLevel Enumeration 517
- TMyLoader Class 574
- TMyLoader.Connection Property 576
- TMyLoader.DuplicateKeys Property 577
- TMyLoader.Options Property 577
- TMyLoader.RowsPerQuery Property 577
- TMyLoaderOption Enumeration 579
- TMyLoaderOptions Set 578
- TMyLogEvent Procedure Reference 611
- TMyMetaData Class 472
- TMyProtocol Enumeration 542
- TMyQuery Class 474
- TMyQuery.FetchAll Property 484
- TMyQuery.LockMode Property 484
- TMyQuery.UpdatingTable Property 484
- TMyRestoreDuplicates Enumeration 533
- TMyScript Class 581
- TMyScript.Connection Property 583
- TMyScript.DataSet Property 584
- TMyScript.UseOptimization Property 584
- TMyServerControl Class 586
- TMyServerControl.AnalyzeTable Method 603
- TMyServerControl.CheckTable Method 604
- TMyServerControl.Connection Property 597
- TMyServerControl.CreateDatabase Method 604
- TMyServerControl.Debug Property 597
- TMyServerControl.DropDatabase Method 605
- TMyServerControl.Flush Method 605
- TMyServerControl.GetServiceNames Method 605
- TMyServerControl.KillProcess Method 606
- TMyServerControl.OptimizeTable Method 606
- TMyServerControl.RepairTable Method 607
- TMyServerControl.ServiceStart Method 607
- TMyServerControl.ServiceStatus Method 608
- TMyServerControl.ServiceStop Method 608
- TMyServerControl.ShowProcessList Method 608
- TMyServerControl.ShowStatus Method 609
- TMyServerControl.ShowVariables Method 609
- TMyServerControl.TableNames Property 597
- TMyServerControl.Variables Property(Indexer) 598
- TMySQLMonitor Class 614
- TMyStoredProc Class 485
- TMyStoredProc.LockMode Property 495
- TMyStoredProc.UpdatingTable Property 496
- TMyTable Class 496
- TMyTable.FetchAll Property 506
- TMyTable.LockMode Property 507
- TMyTable.OrderFields Property 507
- TMyTable.TableName Property 507
- TMyTableMsgEvent Procedure Reference 531
- TMyTableOptions Class 508
- TMyTableOptions.HandlerIndex Property 512
- TMyTableOptions.UseHandler Property 512
- TMyTransaction Class 513
- TMyUpdateExecuteEvent Procedure Reference 515
- TMyUpdateSQL Class 513
- TObjectType Class 342
- TObjectType.AttributeByName Method 345
- TObjectType.AttributeCount Property 343



TObjectType.Attributes Property(Indexer) 343  
TObjectType.DataType Property 344  
TObjectType.FindAttribute Method 345  
TObjectType.Size Property 344  
TOnErrorEvent Procedure Reference 165  
TOnSQLEvent Procedure Reference 174  
TPoolingOptions Class 310  
TPoolingOptions.ConnectionLifetime Property 311  
TPoolingOptions.MaxPoolSize Property 311  
TPoolingOptions.MinPoolSize Property 312  
TPoolingOptions.Validate Property 312  
TProxyOptions Class 120  
TProxyOptions.Hostname Property 121  
TProxyOptions.Password Property 121  
TProxyOptions.Port Property 121  
TProxyOptions.Username Property 121  
TraceFlags Property 171  
TRefreshOption Enumeration 318  
TRefreshOptions Set 315  
TRetryMode Enumeration 318  
TriggerNames Property 557  
TrimFixedChar Property 470  
Truncate Method 339  
TSharedObject Class 346  
TSharedObject.AddRef Method 347  
TSharedObject.RefCount Property 347  
TSharedObject.Release Method 348  
TSortType Enumeration 351  
TUpdateExecuteEvent Procedure Reference 315  
TUpdateReckKind Enumeration 352  
TUpdateReckKinds Set 349  
TVirtualTable Class 616  
TVirtualTable.AddField Method 621  
TVirtualTable.Assign Method 622  
TVirtualTable.Clear Method 622  
TVirtualTable.DeleteField Method 622  
TVirtualTable.DeleteFields Method 623  
TVirtualTable.LoadFromFile Method 623  
TVirtualTable.LoadFromStream Method 624  
TVirtualTable.Options Property 619  
TVirtualTable.SaveToFile Method 624  
TVirtualTable.SaveToStream Method 624  
TVirtualTableOption Enumeration 627

TVirtualTableOptions Set 626

## - U -

ukDelete 352  
ukInsert 352  
ukUpdate 352  
UniDirectional Property 223  
UnLock Method 237  
UnLockTable Method 411  
UnPrepare Method  
    TCustomDASQL 252  
    TMemDataSet 368  
Update Method 324  
UpdateAllFields Property 269  
UpdateBatchSize Property 269  
UpdateRecordTypes Property 358  
UpdateResult Method 369  
UpdatesPending Property 358  
UpdateStatus Method 369  
Updating Data with MyDAC Dataset Components 40  
UpdatingTable Property  
    TMyQuery 484  
    TMyStoredProc 496  
Url Property 119  
UseDelayedIns Property 560  
UseExtSyntax Property 560  
UseHandler Property 512  
UseOptimization Property 584  
Username Property  
    TCustomDAConnection 194  
    THttpOptions 120  
    TProxyOptions 121  
UsernameLabel Property 187  
UseUnicode Property 391  
Using Several DAC Products in One IDE 65

## - V -

Validate Property 312  
Value Property  
    TDAParam 295  
    TMacro 306  
Variables Property(Indexer) 598  
Version Property 537  
VirtualTable Unit Members 615

voPersistentData 627

voStored 627

## **- W -**

WaitExecuting Method 252

What's New 12

Working in an Unstable Network 52

Write Method 339

Writing GUI Applications with MyDAC 73