

Table of Contents

Part I What's New	1
Part II General Information	13
1 Overview	14
2 Features	17
3 Requirements	21
4 Compatibility	22
5 Using Several DAC Products in One IDE	28
6 Component List	29
7 Hierarchy Chart	31
8 Editions	33
9 Licensing	36
10 Getting Support	39
11 Frequently Asked Questions	41
Part III Getting Started	48
1 Installation	54
2 Migrating from BDE and IBX	57
3 Connecting to InterBase and Firebird	58
4 Deleting Data From Tables	63
5 Creating Database Objects	66
6 Inserting Data Into Tables	72
7 Retrieving Data From Tables	77
8 Modifying Data in Tables	80
9 Using Stored Procedures	83
10 Using Stored Procedures with Result Sets	89
11 Demo Projects	92
12 Deployment	98
Part IV Using IBDAC	99
1 Updating Data with IBDAC Dataset Components	100
2 Master/Detail Relationships	102
3 Automatic Key Field Value Generation	105
4 Data Type Mapping	105
5 OTW Network Encryption	112
6 Data Encryption	114
7 Working in an Unstable Network	116

8	Disconnected Mode	117
9	Batch Operations	119
10	Increasing Performance	123
11	Macros	125
12	DataSet Manager	126
13	Connection Pooling	132
14	BLOB Data Types	135
15	Unicode Character Data	137
16	ARRAY Data Type	139
17	TIBCQuery Component	141
18	DBMonitor	142
19	Writing GUI Applications with IBDAC	142
20	Compatibility with Previous Versions	143
21	64-bit Development with Embarcadero RAD Studio XE2	144
22	Database Specific Aspects of 64-bit Development	150
Part V Reference		150
1	CRAccess	152
	Classes	153
	TCRCursor Class	154
	Members	154
	Types	155
	TBeforeFetchProc Procedure Reference	155
	Enumerations	156
	TCRIsolationLevel Enumeration	156
	TCRTransactionAction Enumeration	157
	TCursorState Enumeration	158
2	CRBatchMove	158
	Classes	159
	TCRBatchMove Class	160
	Members	160
	Properties	162
	AbortOnKeyViol Property	164
	AbortOnProblem Property	164
	ChangedCount Property	165
	CommitCount Property	165
	Destination Property	166
	FieldMappingMode Property	166
	KeyViolCount Property	167
	Mappings Property	167
	Mode Property	168
	MovedCount Property	169
	ProblemCount Property	169
	RecordCount Property	170
	Source Property	170
	Methods	171
	Execute Method	171

Events	172
OnBatchMoveProgress Event	172
Types	173
TCRBatchMoveProgressEvent Procedure Reference	173
Enumerations	173
TCRBatchMode Enumeration	174
TCRFieldMappingMode Enumeration	175
3 CREncryption	175
Classes	176
TCREncryptor Class	176
Members	177
Properties	177
DataHeader Property	178
EncryptionAlgorithm Property	179
HashAlgorithm Property	179
InvalidHashAction Property	180
Password Property	180
Methods	181
SetKey Method	181
Enumerations	182
TCREncDataHeader Enumeration	183
TCREncryptionAlgorithm Enumeration	183
TCRHashAlgorithm Enumeration	184
TCRInvalidHashAction Enumeration	184
4 CRVio	185
Enumerations	185
TIPVersion Enumeration	185
5 DAAlerter	186
Classes	187
TDAAlerter Class	187
Members	188
Properties	188
Active Property	189
AutoRegister Property	190
Connection Property	190
Methods	191
SendEvent Method	191
Start Method	192
Stop Method	192
Events	193
OnError Event	193
Types	194
TAlerterErrorEvent Procedure Reference	194
6 DADump	195
Classes	195
TDADump Class	196
Members	196
Properties	198
Connection Property	199
Debug Property	199
Options Property	200
SQL Property	201
TableNames Property	202

Methods	202
Backup Method	203
BackupQuery Method	204
BackupToFile Method	204
BackupToStream Method	205
Restore Method	206
RestoreFromFile Method	207
RestoreFromStream Method	207
Events	208
OnBackupProgress Event	209
OnError Event	210
OnRestoreProgress Event	210
TDADumpOptions Class	211
Members	211
Properties	212
AddDrop Property	213
CompleteInsert Property	213
GenerateHeader Property	214
QuoteNames Property	214
Types	215
TDABackupProgressEvent Procedure Reference	215
TDARestoreProgressEvent Procedure Reference	216
7 DALoader	216
Classes	217
TDAColumn Class	217
Members	218
Properties	218
FieldType Property	219
Name Property	219
TDAColumns Class	220
Members	221
Properties	221
Items Property(Indexer)	221
TDALoader Class	222
Members	223
Properties	224
Columns Property	224
Connection Property	225
TableName Property	225
Methods	226
CreateColumns Method	227
Load Method	227
LoadFromDataSet Method	228
PutColumnData Method	228
PutColumnData Method	229
PutColumnData Method	230
Events	230
OnGetColumnData Event	231
OnProgress Event	232
OnPutData Event	233
TDALoaderOptions Class	233
Members	234
Properties	234
UseBlankValues Property	235

Types	235
TDAPutDataEvent Procedure Reference	235
TGetColumnDataEvent Procedure Reference	236
TLoaderProgressEvent Procedure Reference	237
8 DAScript	237
Classes	238
TDA Script Class	239
Members	239
Properties	241
Connection Property	243
DataSet Property	243
Debug Property	244
Delimiter Property	244
EndLine Property	245
EndOffset Property	245
EndPos Property	246
Macros Property	246
SQL Property	247
StartLine Property	247
StartOffset Property	248
StartPos Property	248
Statements Property	249
Methods	250
BreakExec Method	251
ErrorOffset Method	251
Execute Method	252
ExecuteFile Method	252
ExecuteNext Method	253
ExecuteStream Method	253
FindMacro Method	254
MacroByName Method	255
Events	256
AfterExecute Event	256
BeforeExecute Event	257
OnError Event	257
TDA Statement Class	258
Members	258
Properties	259
EndLine Property	260
EndOffset Property	261
EndPos Property	261
Omit Property	262
Params Property	262
Script Property	263
SQL Property	263
StartLine Property	264
StartOffset Property	264
StartPos Property	264
Methods	265
Execute Method	265
TDA Statements Class	266
Members	266
Properties	267
Items Property(Indexer)	267

Types	268
TAfterStatementExecuteEvent Procedure Reference	268
TBeforeStatementExecuteEvent Procedure Reference	269
TOnErrorEvent Procedure Reference	269
Enumerations	270
TErrorAction Enumeration	270
9 DASQLMonitor	271
Classes	272
TCustomDASQLMonitor Class	272
Members	273
Properties	274
Active Property	274
DBMonitorOptions Property	275
Options Property	275
TraceFlags Property	276
Events	276
OnSQL Event	277
TDBMonitorOptions Class	277
Members	278
Properties	278
Host Property	279
Port Property	280
ReconnectTimeout Property	280
SendTimeout Property	281
Types	281
TDATraceFlags Set	282
TMonitorOptions Set	282
TOnSQLEvent Procedure Reference	282
Enumerations	283
TDATraceFlag Enumeration	283
TMonitorOption Enumeration	284
10 DBAccess	285
Classes	288
EDAError Class	290
Members	290
Properties	291
Component Property	291
ErrorCode Property	292
TCRDataSource Class	292
Members	293
TCustomConnectDialog Class	293
Members	294
Properties	295
CancelButton Property	296
Caption Property	296
ConnectButton Property	297
DialogClass Property	297
LabelSet Property	298
PasswordLabel Property	298
Retries Property	299
SavePassword Property	299
ServerLabel Property	300
StoreLogInfo Property	300

UsernameLabel Property	301
Methods	301
Execute Method	302
GetServerList Method	302
TCustomDAConnection Class	303
Members	303
Properties	306
ConnectDialog Property	307
ConnectString Property	307
ConvertEOL Property	309
InTransaction Property	309
LoginPrompt Property	310
Options Property	310
Password Property	311
Pooling Property	312
PoolingOptions Property	313
Server Property	314
Username Property	315
Methods	315
ApplyUpdates Method	317
ApplyUpdates Method	317
ApplyUpdates Method	318
Commit Method	318
Connect Method	319
CreateSQL Method	320
Disconnect Method	321
ExecProc Method	321
ExecProcEx Method	323
ExecSQL Method	325
ExecSQLEx Method	326
GetDatabaseNames Method	327
GetKeyFieldNames Method	328
GetStoredProcNames Method	328
GetTableNames Method	329
MonitorMessage Method	330
Ping Method	331
RemoveFromPool Method	331
Rollback Method	332
StartTransaction Method	333
Events	333
OnConnectionLost Event	334
OnError Event	334
TCustomDADataSet Class	335
Members	336
Properties	343
BaseSQL Property	347
Conditions Property	348
Connection Property	348
DataTypeMap Property	349
Debug Property	349
DetailFields Property	350
Disconnected Property	351
FetchRows Property	351
FilterSQL Property	352

FinalSQL Property.....	352
IsQuery Property.....	353
KeyFields Property.....	354
MacroCount Property.....	355
Macros Property.....	355
MasterFields Property.....	356
MasterSource Property.....	357
Options Property.....	358
ParamCheck Property.....	360
ParamCount Property.....	361
Params Property.....	361
ReadOnly Property.....	362
RefreshOptions Property.....	363
RowsAffected Property.....	363
SQL Property.....	364
SQLDelete Property.....	365
SQLInsert Property.....	365
SQLLock Property.....	366
SQLRecCount Property.....	367
SQLRefresh Property.....	368
SQLUpdate Property.....	369
UniDirectional Property.....	370
Methods.....	371
AddWhere Method.....	375
BreakExec Method.....	375
CreateBlobStream Method.....	376
DeleteWhere Method.....	377
Execute Method.....	377
Execute Method.....	378
Execute Method.....	379
Executing Method.....	379
Fetched Method.....	380
Fetching Method.....	380
FetchingAll Method.....	381
FindKey Method.....	382
FindMacro Method.....	382
FindNearest Method.....	383
FindParam Method.....	384
GetData Type Method.....	385
GetFieldObject Method.....	385
GetFieldPrecision Method.....	386
GetFieldScale Method.....	387
GetKeyFieldNames Method.....	387
GetOrderBy Method.....	388
GotoCurrent Method.....	389
Lock Method.....	389
MacroByName Method.....	390
ParamByName Method.....	391
Prepare Method.....	392
RefreshRecord Method.....	393
RestoreSQL Method.....	393
SaveSQL Method.....	394
SetOrderBy Method.....	395
SQLSaved Method.....	395

UnLock Method.....	396
Events	396
AfterExecute Event.....	397
AfterFetch Event.....	398
AfterUpdateExecute Event.....	398
BeforeFetch Event.....	399
BeforeUpdateExecute Event.....	399
TCustomDA SQL Class.....	400
Members	400
Properties	403
ChangeCursor Property.....	404
Connection Property.....	405
Debug Property.....	405
FinalSQL Property.....	406
MacroCount Property.....	406
Macros Property.....	407
ParamCheck Property.....	408
ParamCount Property.....	408
Params Property.....	409
ParamValues Property(Indexer).....	410
Prepared Property.....	411
RowsAffected Property.....	411
SQL Property	412
Methods	413
BreakExec Method.....	414
Execute Method.....	414
Execute Method.....	415
Execute Method.....	415
Executing Method.....	416
FindMacro Method.....	416
FindParam Method.....	417
MacroByName Method.....	418
ParamByName Method.....	419
Prepare Method.....	419
UnPrepare Method.....	420
WaitExecuting Method.....	421
Events	421
AfterExecute Event.....	422
TCustomDAUpdateSQL Class.....	422
Members	423
Properties	424
DataSet Property.....	426
DeleteObject Property.....	426
DeleteSQL Property.....	427
InsertObject Property.....	427
InsertSQL Property.....	428
LockObject Property.....	428
LockSQL Property.....	429
ModifyObject Property.....	429
ModifySQL Property.....	430
RefreshObject Property.....	430
RefreshSQL Property.....	431
SQL Property(Indexer).....	432
Methods	432

Apply Method	433
ExecSQL Method.....	434
TDACondition Class.....	434
Members	435
Properties	436
Enabled Property.....	436
Name Property	437
Value Property.....	437
Methods	437
Disable Method.....	438
Enable Method.....	438
TDAConditions Class.....	438
Members	439
Properties	440
Condition Property(Indexer).....	441
Enabled Property.....	442
Items Property(Indexer).....	442
Text Property	442
WhereSQL Property.....	443
Methods	443
Add Method	444
Add Method	445
Add Method	445
Delete Method.....	446
Disable Method.....	446
Enable Method.....	447
Find Method	447
Get Method	447
IndexOf Method.....	448
Remove Method.....	448
TDAConnectionOptions Class.....	449
Members	449
Properties	450
Allow ImplicitConnect Property	451
DefaultSortType Property.....	452
DisconnectedMode Property.....	452
KeepDesignConnected Property.....	453
LocalFailover Property.....	453
TDAConnectionSSLOptions Class.....	454
Members	454
Properties	455
CACert Property.....	455
Cert Property	456
CipherList Property	456
Key Property	457
TDADataSetOptions Class.....	457
Members	457
Properties	460
AutoPrepare Property.....	463
CacheCalcFields Property.....	463
CompressBlobMode Property	464
DefaultValues Property.....	464
DetailDelay Property.....	465
FieldsOrigin Property.....	465

FlatBuffers Property.....	466
InsertAllSetFields Property.....	466
LocalMasterDetail Property.....	467
LongStrings Property.....	467
MasterFieldsNullable Property.....	468
NumberRange Property.....	468
QueryRecCount Property.....	469
QuoteNames Property.....	469
RemoveOnRefresh Property.....	470
RequiredFields Property.....	470
ReturnParams Property.....	471
SetFieldsReadOnly Property.....	471
StrictUpdate Property.....	472
TrimFixedChar Property.....	472
UpdateAllFields Property.....	473
UpdateBatchSize Property.....	473
TDAEncryption Class.....	474
Members.....	474
Properties.....	475
Encryptor Property.....	475
Fields Property.....	476
TDAMapRule Class.....	476
Members.....	477
Properties.....	478
DBLengthMax Property.....	479
DBLengthMin Property.....	479
DBScaleMax Property.....	480
DBScaleMin Property.....	480
DBType Property.....	481
FieldLength Property.....	481
FieldName Property.....	482
FieldScale Property.....	482
FieldType Property.....	483
IgnoreErrors Property.....	483
TDAMapRules Class.....	483
Members.....	484
Properties.....	484
IgnoreInvalidRules Property.....	485
TDAMetaData Class.....	486
Members.....	487
Properties.....	490
Connection Property.....	491
MetaDataKind Property.....	492
Restrictions Property.....	493
Methods.....	494
GetMetaDataKinds Method.....	496
GetRestrictions Method.....	497
TDAParam Class.....	497
Members.....	498
Properties.....	500
AsBlob Property.....	501
AsBlobRef Property.....	502
AsFloat Property.....	502
AsInteger Property.....	503

AsLargeInt Property.....	503
AsMemo Property.....	504
AsMemoRef Property.....	504
AsSQLTimeStamp Property.....	505
AsString Property.....	505
AsWideString Property.....	506
DataType Property.....	506
IsNull Property.....	507
ParamType Property.....	507
Size Property.....	508
Value Property.....	508
Methods.....	509
AssignField Method.....	510
AssignFieldValue Method.....	510
LoadFromFile Method.....	511
LoadFromStream Method.....	512
SetBlobData Method.....	512
SetBlobData Method.....	513
SetBlobData Method.....	513
TDAParams Class.....	514
Members.....	514
Properties.....	515
Items Property(Indexer).....	515
Methods.....	516
FindParam Method.....	517
ParamByName Method.....	517
TDATransaction Class.....	518
Members.....	519
Properties.....	520
Active Property.....	520
DefaultCloseAction Property.....	521
Methods.....	521
Commit Method.....	522
Rollback Method.....	523
StartTransaction Method.....	523
Events.....	524
OnCommit Event.....	525
OnCommitRetaining Event.....	526
OnError Event.....	526
OnRollback Event.....	527
OnRollbackRetaining Event.....	528
TMacro Class.....	528
Members.....	529
Properties.....	530
Active Property.....	531
AsDateTime Property.....	531
AsFloat Property.....	532
AsInteger Property.....	532
AsString Property.....	532
Name Property.....	533
Value Property.....	533
TMacros Class.....	534
Members.....	534
Properties.....	535

Items Property(Indexer).....	536
Methods	536
AssignValues Method.....	537
Expand Method.....	537
FindMacro Method.....	538
IsEqual Method.....	539
MacroByName Method.....	539
Scan Method	540
TPoolingOptions Class.....	540
Members	541
Properties	542
ConnectionLifetime Property.....	542
MaxPoolSize Property.....	543
MinPoolSize Property.....	543
PoolId Property.....	544
Validate Property.....	544
TSmartFetchOptions Class.....	545
Members	545
Properties	546
Enabled Property.....	546
LiveBlock Property.....	547
PrefetchedFields Property.....	547
SQLGetKeyValues Property.....	548
Types	548
TAfterExecuteEvent Procedure Reference.....	549
TAfterFetchEvent Procedure Reference.....	550
TBeforeFetchEvent Procedure Reference.....	550
TConnectionLostEvent Procedure Reference.....	551
TDAConnectionErrorEvent Procedure Reference.....	551
TDATransactionErrorEvent Procedure Reference.....	552
TRefreshOptions Set.....	552
TUpdateExecuteEvent Procedure Reference.....	553
Enumerations	553
TLabelSet Enumeration.....	554
TLockMode Enumeration.....	555
TRefreshOption Enumeration.....	555
TRetryMode Enumeration.....	556
Variables	556
ChangeCursor Variable.....	557
11 IBC	557
Classes	560
TCustomIBCDataset Class.....	562
Members	563
Properties	573
AutoCommit Property.....	578
Connection Property.....	579
Cursor Property.....	579
DMLRefresh Property.....	580
Encryption Property.....	580
ExplainPlan Property.....	581
FetchAll Property.....	581
GeneratorMode Property.....	582
GeneratorStep Property.....	582
Handle Property.....	583

IsQuery Property.....	583
KeyGenerator Property.....	584
LockMode Property.....	584
Options Property.....	585
Plan Property	587
Row sDeleted Property.....	587
Row sFetched Property.....	587
Row sInserted Property.....	588
Row sUpdated Property.....	588
SmartFetch Property.....	589
SQLType Property	589
Transaction Property.....	590
UpdateObject Property.....	590
UpdateTransaction Property.....	591
Methods	591
CreateProcCall Method.....	596
FindParam Method.....	596
GetArray Method.....	597
GetBlob Method.....	598
ParamByName Method.....	599
TCustomIBCQuery Class.....	599
Members	601
TCustomIBCTable Class.....	610
Members	612
Properties	621
Exists Property.....	627
Methods	627
DeleteTable Method.....	632
EmptyTable Method.....	632
TIBCArray Class.....	633
Members	633
Properties	637
ItemType Property.....	638
TIBCArrayField Class.....	639
Members	639
Properties	640
AsArray Property.....	640
TIBConnection Class.....	641
Members	642
Properties	646
AutoCommit Property	649
ClientLibrary Property	650
Connected Property.....	650
ConnectPrompt Property.....	651
Database Property.....	652
DatabaseInfo Property.....	653
DBSQLDialect Property.....	653
Debug Property.....	654
DefaultTransaction Property.....	654
Handle Property	655
LastError Property.....	655
Options Property.....	656
Params Property.....	657
Passw ord Property.....	658

Port Property	659
Server Property	659
SQL Property	660
SQLDialect Property.....	660
SSLOptions Property.....	661
TransactionCount Property.....	661
Transactions Property(Indexer).....	662
Username Property.....	662
Methods	663
AddTransaction Method.....	665
AssignConnect Method.....	666
CommitRetaining Method.....	667
CreateDatabase Method.....	667
DropDatabase Method.....	668
FindDefaultTransaction Method.....	668
ParamByName Method.....	669
RemoveTransaction Method.....	670
RollbackRetaining Method.....	670
TIBConnectionOptions Class.....	671
Members	671
Properties	673
CharLength Property.....	675
Charset Property.....	675
EnableBCD Property.....	676
EnableFMTBCD Property.....	676
EnableMemos Property.....	676
IPVersion Property.....	677
NoDBTriggers Property.....	678
Protocol Property.....	678
Role Property	679
TrustedAuthentication Property.....	679
UseUnicode Property.....	680
TIBDataSetOptions Class.....	680
Members	681
Properties	684
AutoClose Property.....	689
BooleanDomainFields Property.....	689
CacheArrays Property.....	690
CacheBlobs Property.....	690
ComplexArrayFields Property.....	691
DefaultValues Property.....	691
DeferredArrayRead Property.....	692
DeferredBlobRead Property.....	692
DescribeParams Property.....	693
ExtendedFieldsInfo Property.....	693
FieldsAsString Property.....	694
FullRefresh Property.....	694
PrepareUpdateSQL Property.....	695
QueryRow sAffected Property.....	695
SetDomainNames Property.....	696
SetEmptyStrToNull Property.....	696
StreamedBlobs Property.....	696
StrictUpdate Property.....	697
TIBDataSource Class.....	697

Members	698
TIBCDbKeyField Class.....	698
Members	699
TIBCEncryptor Class.....	699
Members	699
TIBCMetaData Class.....	700
Members	701
Properties	704
Transaction Property	706
TIBCPParam Class.....	706
Members	707
Properties	709
AsArray Property.....	711
AsIbBlob Property.....	711
Methods	712
SetBlobData Method.....	713
TIBCPParams Class.....	713
Members	714
Properties	715
Items Property(Indexer).....	715
Methods	716
FindParam Method.....	716
ParamByName Method.....	717
TIBCQuery Class.....	718
Members	719
Properties	729
FetchAll Property.....	735
LockMode Property.....	735
UpdatingTable Property.....	736
TIBCSQL Class.....	737
Members	737
Properties	740
Connection Property.....	742
DescribeParams Property.....	743
ExplainPlan Property.....	743
Handle Property.....	744
Params Property.....	744
Plan Property	745
SQLType Property.....	745
Transaction Property.....	746
Methods	746
CreateProcCall Method.....	747
ExecuteNext Method.....	748
FindParam Method.....	749
ParamByName Method.....	749
TIBCSSLConnectionOptions Class.....	750
Members	750
Properties	751
ClientCertFile Property.....	752
ClientPassPhrase Property.....	752
ClientPassPhraseFile Property.....	753
Enabled Property.....	753
ServerPublicFile Property.....	754
ServerPublicPath Property.....	754

TIBCStoredProc Class.....	755
Members	756
Properties	765
LockMode Property.....	771
StoredProcName Property.....	772
Methods	772
ExecProc Method.....	777
Prepare Method.....	777
PrepareSQL Method.....	778
TIBCTable Class.....	778
Members	779
Properties	789
FetchAll Property.....	795
LockMode Property.....	796
TableName Property.....	797
TIBCTransaction Class.....	797
Members	798
Properties	800
Active Property.....	802
Connections Property(Indexer).....	802
ConnectionsCount Property.....	803
DefaultCloseAction Property.....	803
DefaultConnection Property.....	804
Handle Property.....	804
IsolationLevel Property.....	805
Params Property.....	805
Methods	806
AddConnection Method.....	807
Commit Method.....	808
CommitRetaining Method.....	808
FindDefaultConnection Method.....	809
ReleaseSavepoint Method.....	809
RemoveConnection Method.....	810
Rollback Method.....	810
RollbackRetaining Method.....	811
RollbackSavepoint Method.....	811
StartSavepoint Method.....	812
Events	813
OnError Event.....	814
TIBCUpdateSQL Class.....	814
Members	815
Types	816
TIBCTransactionErrorEvent Procedure Reference.....	817
Enumerations	817
TGeneratorMode Enumeration.....	818
TIBCProtocol Enumeration.....	818
TIBCTransactionAction Enumeration.....	819
Variables	819
Connections Variable.....	820
DefConnection Variable.....	820
UseDefConnection Variable.....	821
Constants	821
IBDACVersion Constant.....	821
12 IBCAdmin	822

Classes	824
TCustomIBCSERVICE Class.....	826
Members	827
Properties	828
Active Property.....	829
Handle Property.....	829
LoginPrompt Property.....	829
Params Property.....	830
Protocol Property.....	830
Server Property.....	831
ServiceParamBySPB Property (Indexer).....	831
Methods	832
Attach Method.....	832
Detach Method.....	833
ServiceStart Method.....	833
Events	833
OnAttach Event.....	834
TIBCBACKUPRESTORESERVICE Class.....	834
Members	835
Properties	837
BackupFile Property.....	838
NBackupLevel Property.....	838
NBackupOptions Property.....	839
UseNBackup Property.....	839
Verbose Property.....	840
TIBCBACKUPSERVICE Class.....	840
Members	841
Properties	843
BackupFile Property.....	844
BlockingFactor Property.....	845
Database Property.....	845
Options Property.....	846
Verbose Property.....	846
TIBCCONFIGPARAMS Class.....	847
Members	847
Properties	848
BaseLocation Property.....	848
LockFileLocation Property.....	849
MessageFileLocation Property.....	849
SecurityDatabaseLocation Property.....	850
TIBCCONFIGSERVICE Class.....	850
Members	851
Properties	853
Connection Property.....	854
Database Property.....	855
JournalInformation Property.....	855
Transaction Property.....	856
Methods	856
ActivateShadow Method.....	858
AlterJournal Method.....	858
BringDatabaseOnline Method.....	859
CreateJournal Method.....	859
CreateJournalArchive Method.....	860
DisableFlush Method.....	860

DropJournal Method.....	861
DropJournalArchive Method.....	861
FlushDatabase Method.....	862
GetJournalInformation Method.....	862
ReclaimMemory Method.....	863
ServiceStart Method.....	863
SetAsyncMode Method.....	864
SetDBSqlDialect Method.....	864
SetFlushInterval Method.....	865
SetGroupCommit Method.....	865
SetLingerInterval Method.....	866
SetPageBuffers Method.....	867
SetReadOnly Method.....	867
SetReclaimInterval Method.....	868
SetReserveSpace Method.....	869
SetSleepInterval Method.....	869
ShutdownDatabase Method.....	870
SleepDatabase Method.....	871
TIBControlAndQueryService Class.....	871
Members.....	872
Properties.....	873
BufferSize Property.....	874
Eof Property.....	875
Methods.....	875
GetNextChunk Method.....	876
GetNextLine Method.....	876
TIBDatabaseInfo Class.....	877
Members.....	877
Properties.....	877
DbName Property.....	878
NoOfAttachments Property.....	878
TIBJournalInformation Class.....	879
Members.....	879
Properties.....	880
CheckpointInterval Property.....	881
CheckpointLength Property.....	882
Directory Property.....	882
HasArchive Property.....	883
HasJournal Property.....	883
PageCache Property.....	884
PageLength Property.....	884
PageSize Property.....	884
TimestampName Property.....	885
TIBLicenseInfo Class.....	886
Members.....	886
Properties.....	886
Desc Property.....	887
Id Property.....	888
Key Property.....	888
LicensedUsers Property.....	888
TIBLicenseMaskInfo Class.....	889
Members.....	889
Properties.....	890
CapabilityMask Property.....	891

LicenseMask Property.....	891
TIBCLicensingService Class.....	892
Members.....	892
Properties.....	894
Action Property.....	895
ID Property.....	895
Key Property.....	896
Methods.....	896
AddLicense Method.....	897
RemoveLicense Method.....	898
TIBCLimboTransactionInfo Class.....	898
Members.....	899
Properties.....	899
Action Property.....	900
Advise Property.....	901
HostSite Property.....	901
ID Property.....	902
MultiDatabase Property.....	902
RemoteDatabasePath Property.....	902
RemoteSite Property.....	903
State Property.....	903
TIBCLogService Class.....	904
Members.....	904
TIBCRestoreService Class.....	906
Members.....	906
Properties.....	908
BackupFile Property.....	910
Database Property.....	910
Options Property.....	911
PageBuffers Property.....	911
PageSize Property.....	912
Verbose Property.....	912
TIBCSecurityService Class.....	912
Members.....	913
Properties.....	915
SecurityAction Property.....	916
UserDatabase Property.....	917
UserInfo Property.....	917
UserInfos Property (Indexer).....	918
UserInfosCount Property.....	918
Methods.....	919
AddUser Method.....	920
DeleteUser Method.....	920
DisplayUser Method.....	921
DisplayUsers Method.....	921
EnableEUA Method.....	922
ModifyUser Method.....	923
SuspendEUA Method.....	923
TIBCServerProperties Class.....	924
Members.....	924
Properties.....	926
AliasCount Property.....	927
ConfigParams Property.....	928
DatabaseInfo Property.....	928

LicenseInfo Property.....	929
LicenseMaskInfo Property.....	929
VersionInfo Property.....	930
Methods	930
AddAlias Method.....	931
DeleteAlias Method.....	932
Fetch Method	933
FetchAliasInfo Method.....	933
FetchConfigParams Method.....	934
FetchDatabaseInfo Method.....	934
FetchLicenseInfo Method.....	935
FetchLicenseMaskInfo Method.....	935
FetchVersionInfo Method.....	936
TIBCStatisticalService Class.....	936
Members	937
Properties	938
Database Property.....	940
Options Property.....	940
TableNames Property.....	941
TIBCTraceService Class.....	941
Members	942
Properties	943
Config Property.....	945
SessionName Property.....	945
Methods	946
ListTraceSessions Method.....	947
ResumeTrace Method.....	947
StartTrace Method.....	948
StopTrace Method.....	948
SuspendTrace Method.....	949
TIBCUserInfo Class.....	950
Members	950
Properties	951
ActiveUser Property.....	952
DefaultRole Property.....	952
Description Property.....	953
FirstName Property.....	953
GroupID Property.....	954
GroupName Property.....	954
LastName Property.....	954
MiddleName Property.....	955
Password Property.....	955
SQLRole Property.....	956
SystemUserName Property.....	956
UserID Property.....	957
UserName Property.....	957
TIBCValidationService Class.....	957
Members	958
Properties	960
Database Property.....	961
GlobalAction Property.....	962
LimboTransactionInfo Property (Indexer).....	962
LimboTransactionInfoCount Property.....	963
Options Property.....	963

RecoverTwoPhaseGlobal Property	964
Methods	964
FetchLimboTransactionInfo Method.....	965
FixLimboTransactionErrors Method.....	966
SweepDatabase Method.....	966
TIBCVersionInfo Class	967
Members	967
Properties	967
ServerImplementation Property.....	968
ServerVersion Property.....	969
ServiceVersion Property.....	969
Types	969
TIBCBackupOptions Set.....	970
TIBCNBackupOptions Set.....	970
TIBCRestoreOptions Set.....	971
TIBCStatOptions Set.....	971
TIBCValidateOptions Set.....	971
Enumerations	972
TIBCBackupOption Enumeration.....	972
TIBCLicensingAction Enumeration.....	973
TIBCNBackupOption Enumeration.....	974
TIBCRestoreOption Enumeration.....	974
TIBCSecurityAction Enumeration.....	975
TIBCStatOption Enumeration.....	976
TIBCTransactionAdvise Enumeration.....	977
TIBCTransactionGlobalAction Enumeration.....	977
TIBCTransactionState Enumeration.....	978
TIBCValidateOption Enumeration.....	978
13 IBCAlerter	979
Classes	979
TIBCAlerter Class.....	980
Members	981
Properties	982
AutoCommit Property.....	983
Connection Property.....	983
Events Property.....	984
Transaction Property.....	984
Methods	984
SendEvent Method.....	985
Events	986
OnEvent Event.....	986
Types	987
TIBCAlerterEvent Procedure Reference.....	987
14 IBCArray	988
Classes	988
TCustomIBCArray Class.....	988
Members	989
Properties	992
ArrayDimensions Property.....	994
ArrayHighBound Property(Indexer).....	994
ArrayID Property.....	995
ArrayLowBound Property(Indexer).....	995
ArraySize Property.....	996

AsString Property.....	996
Cached Property.....	997
CachedDimensions Property.....	997
CachedHighBound Property(Indexer).....	998
CachedLow Bound Property(Indexer).....	998
CachedSize Property.....	999
ColumnName Property.....	999
DbHandle Property.....	1000
IsNull Property.....	1000
Items Property.....	1001
ItemScale Property.....	1001
ItemSize Property.....	1002
Modified Property.....	1002
TableName Property.....	1003
TrHandle Property.....	1003
Methods	1004
Assign Method.....	1006
ClearArray Method.....	1006
CreateTemporaryArray Method.....	1007
GetArrayInfo Method.....	1007
GetItemAsDateTime Method.....	1008
GetItemAsFloat Method.....	1009
GetItemAsInteger Method.....	1009
GetItemAsSmallInt Method.....	1010
GetItemAsString Method.....	1011
GetItemAsWideString Method.....	1012
GetItemsSlice Method.....	1012
GetItemValue Method.....	1013
ReadArray Method.....	1014
ReadArrayItem Method.....	1014
ReadArraySlice Method.....	1015
SetItemAsDateTime Method.....	1016
SetItemAsFloat Method.....	1016
SetItemAsInteger Method.....	1017
SetItemAsSmallInt Method.....	1018
SetItemAsString Method.....	1019
SetItemAsWideString Method.....	1019
SetItemsSlice Method.....	1020
SetItemValue Method.....	1021
WriteArray Method.....	1021
WriteArraySlice Method.....	1022
15 IBCClasses	1023
Classes	1024
TGDSDatabaseInfo Class.....	1024
Members	1025
Properties	1028
Allocation Property.....	1032
AttachmentID Property.....	1032
BackoutCount Property.....	1032
BaseLevel Property.....	1033
CurLogFileName Property.....	1033
CurLogPartitionOffset Property.....	1034
CurrentMemory Property.....	1034
DBFileName Property.....	1035

DBImplementationClass Property.....	1035
DBImplementationNo Property.....	1035
DBSiteName Property.....	1036
DeleteCount Property.....	1036
ExpungeCount Property.....	1037
Fetches Property.....	1037
ForcedWrites Property.....	1038
InsertCount Property.....	1038
IsRemoteConnect Property.....	1038
LogFile Property.....	1039
Marks Property.....	1039
MaxMemory Property.....	1040
NoReserve Property.....	1040
NumBuffers Property.....	1041
NumWALBuffers Property.....	1041
ODSMajorVersion Property.....	1042
ODSMInorVersion Property.....	1042
PageSize Property.....	1042
PurgeCount Property.....	1043
ReadIdxCount Property.....	1043
ReadOnly Property.....	1044
Reads Property.....	1044
ReadSeqCount Property.....	1045
Sw eepInterval Property.....	1045
UpdateCount Property.....	1046
UserNames Property.....	1046
Version Property.....	1047
WALAverageGroupCommitSize Property.....	1047
WALAverageIOSize Property.....	1047
WALBufferSize Property.....	1048
WALCheckpointLength Property.....	1048
WALCurCheckpointInterval Property.....	1049
WALGroupCommitWaitUSecs Property.....	1049
WALNumCommits Property.....	1050
WALNumIO Property.....	1050
WALPrvCheckpointFilename Property.....	1050
WALPrvCheckpointPartOffset Property.....	1051
Writes Property.....	1051
TIBCBlob Class.....	1052
Members.....	1053
Properties.....	1055
Cached Property.....	1057
CharsetID Property.....	1057
ConversionCharsetID Property.....	1058
ConversionSubType Property.....	1059
DbHandle Property.....	1059
Handle Property.....	1060
ID Property.....	1060
MaxSegmentSize Property.....	1061
NumSegments Property.....	1061
Streamed Property.....	1062
SubType Property.....	1062
TrHandle Property.....	1063
Methods.....	1063

AllocBlob Method.....	1065
CloseBlob Method.....	1065
FreeBlob Method.....	1066
IsInit Method	1066
LengthBlob Method.....	1067
ReadBlob Method.....	1067
WriteBlob Method.....	1068
Enumerations	1068
TIBCSolationLevel Enumeration.....	1068
Routines	1069
Reverse2 Procedure.....	1070
Reverse4 Function.....	1070
XSQLDA_LENGTH Function.....	1071
Variables	1071
IntegerPrecision Variable.....	1072
16 IBConnectionPool	1072
Classes	1072
TIBConnectionPoolManager Class.....	1073
Members	1073
17 IBDataTypeMap	1074
Constants	1075
IBCArray Constant.....	1076
ibcArray Constant.....	1076
ibcBigint Constant.....	1077
ibcBlob Constant.....	1077
ibcBoolean Constant.....	1078
ibcChar Constant.....	1078
ibcCharBin Constant.....	1079
ibcDate Constant.....	1079
ibcDecimal Constant.....	1080
ibcDouble Constant.....	1080
ibcFloat Constant.....	1081
ibcInteger Constant.....	1081
ibcNumeric Constant.....	1082
ibcSmallint Constant.....	1082
ibcText Constant.....	1083
ibcTime Constant.....	1083
ibcTimestamp Constant.....	1083
ibcVarchar Constant.....	1084
ibcVarcharBin Constant.....	1084
18 IBError	1085
Classes	1085
EIBError Class.....	1085
Members	1086
Properties	1087
ErrorNumber Property.....	1087
Sender Property.....	1088
SQLExceptionMsg Property.....	1088
19 IBCLoader	1089
Classes	1089
TIBCLoader Class.....	1090
Members	1091
Properties	1092

Columns Property.....	1092
Options Property.....	1093
TableName Property.....	1093
Events	1094
OnGetColumnData Event.....	1094
OnPutData Event.....	1095
TIBCLoaderOptions Class.....	1095
Members	1096
Properties	1096
InsertMode Property.....	1097
QuoteNames Property.....	1098
RowsPerBatch Property.....	1098
20 IBCScript	1099
Classes	1099
TIBCScript Class.....	1100
Members	1100
Properties	1103
AutoDDL Property.....	1104
Connection Property.....	1105
DataSet Property.....	1105
Params Property.....	1106
Transaction Property.....	1106
21 IBCSQLMonitor	1107
Classes	1107
TIBCSQLMonitor Class.....	1107
Members	1108
22 IbDacVcl	1109
Classes	1109
TIBConnectDialog Class.....	1110
Members	1111
Properties	1112
Connection Property.....	1113
DatabaseLabel Property.....	1114
ProtocolLabel Property.....	1114
Routines	1115
GetIBCDatabaseList Procedure.....	1115
GetIBCDatabaseList Procedure.....	1116
GetIBCServerList Procedure.....	1116
23 MemData	1117
Classes	1119
TAttribute Class.....	1119
Members	1120
Properties	1121
AttributeNo Property.....	1122
DataSize Property.....	1122
DataType Property.....	1123
Length Property.....	1123
ObjectType Property.....	1124
Offset Property.....	1124
Owner Property.....	1125
Scale Property.....	1125
Size Property.....	1126
TBlob Class.....	1126

Members	1127
Properties	1128
AsString Property	1129
AsWideString Property	1130
IsUnicode Property	1130
Size Property	1131
Methods	1131
Assign Method	1132
Clear Method	1133
LoadFromFile Method	1133
LoadFromStream Method	1134
Read Method	1134
SaveToFile Method	1135
SaveToStream Method	1136
Truncate Method	1137
Write Method	1137
TCompressedBlob Class	1138
Members	1139
Properties	1141
Compressed Property	1141
CompressedSize Property	1142
TDBObject Class	1142
Members	1143
TMemData Class	1144
Members	1144
TObjectType Class	1144
Members	1145
Properties	1146
AttributeCount Property	1146
Attributes Property (Indexer)	1147
DataType Property	1147
Size Property	1148
Methods	1148
FindAttribute Method	1149
TSharedObject Class	1150
Members	1150
Properties	1151
RefCount Property	1152
Methods	1152
AddRef Method	1153
Release Method	1153
Types	1154
TLocateExOptions Set	1154
TUpdateRecKinds Set	1154
Enumerations	1155
TCompressBlobMode Enumeration	1155
TConnLostCause Enumeration	1156
TDANumericType Enumeration	1157
TLocateExOption Enumeration	1158
TSortType Enumeration	1159
TUpdateRecKind Enumeration	1159
24 MemDS	1160
Classes	1160
TMemDataSet Class	1161

Members	1161
Properties	1164
CachedUpdates Property.....	1165
IndexFieldNames Property.....	1167
KeyExclusive Property.....	1168
LocalConstraints Property.....	1169
LocalUpdate Property.....	1169
Prepared Property.....	1170
Ranged Property.....	1170
UpdateRecordTypes Property.....	1171
UpdatesPending Property.....	1171
Methods	1172
ApplyRange Method.....	1174
ApplyUpdates Method.....	1175
ApplyUpdates Method.....	1175
ApplyUpdates Method.....	1176
CancelRange Method.....	1177
CancelUpdates Method.....	1178
CommitUpdates Method.....	1179
DeferredPost Method.....	1180
EditRangeEnd Method.....	1180
EditRangeStart Method.....	1181
GetBlob Method.....	1182
GetBlob Method.....	1182
GetBlob Method.....	1183
Locate Method.....	1183
Locate Method.....	1184
Locate Method.....	1184
LocateEx Method.....	1186
LocateEx Method.....	1186
LocateEx Method.....	1187
Prepare Method.....	1188
RestoreUpdates Method.....	1189
RevertRecord Method.....	1190
SaveToXML Method.....	1190
SaveToXML Method.....	1191
SaveToXML Method.....	1191
SetRange Method.....	1192
SetRangeEnd Method.....	1193
SetRangeStart Method.....	1194
UnPrepare Method.....	1195
UpdateResult Method.....	1196
UpdateStatus Method.....	1196
Events	1197
OnUpdateError Event.....	1198
OnUpdateRecord Event.....	1199
Variables	1200
DoNotRaiseExcetionOnUaFail Variable.....	1200
SendDataSetChangeEventAfterOpen Variable.....	1201
25 VirtualDataSet	1201
Classes	1202
TCustomVirtualDataSet Class.....	1202
Members	1203
TVirtualDataSet Class.....	1206

Members	1206
Types	1209
TOnDeleteRecordEvent Procedure Reference.....	1210
TOnGetFieldValueEvent Procedure Reference.....	1210
TOnGetRecordCountEvent Procedure Reference.....	1211
TOnModifyRecordEvent Procedure Reference.....	1212
26 VirtualTable	1212
Classes	1213
TVirtualTable Class.....	1213
Members	1213
Properties	1217
DefaultSortType Property.....	1218
Methods	1218
Assign Method.....	1221
LoadFromFile Method.....	1222
LoadFromStream Method.....	1222

1 What's New

New Features in IBDAC 9.2

- Added support for RAD Studio 12 Athens Release 1
- Added support for Lazarus 3.2

New Features in IBDAC 9.1

- Added support for Lazarus 3.0
- Added support for Firebird 5
- Added support for parallel operations in TIBCBackupService, TIBCRestoreService and TIBCValidationService
- Added support for minor ODS upgrade in TIBCValidationService
- Added support for multiple rows being returned by DML with the RETURNING clause
- Improved compatibility with macOS Sonoma

New Features in IBDAC 9.0

- Added support for RAD Studio 12
- Added support for macOS Sonoma
- Added support for iOS 17
- Added support for Android 13
- Added support for iOS Simulator ARM 64-bit target platform
- Added support for EnableMemos property in FastReport components
- Added support for nested Macros in SQL queries
- Added support Display Format for Aggregate fields
- Added SHA-2(SHA-256, SHA-512) in hash algorithm for encryption

New Features in IBDAC 8.3

- Added support for RAD Studio 11 Alexandria Release 3
- Added support for Lazarus 2.2.6

- Added support for Charset and UseUnicode properties in FastReport components
- Added support for the YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, GETDATE, DATE, TIME, TRIM, TRIMLEFT, TRIMRIGHT statements in TDADataset.Filter
- Added support for the mathematical operations in TDADataset.Filter
- Added support for Aggregate Fields and InternalCalc Fields
- Added ability to restore from file with TEncoding via the Dump component
- Improved work with alias
- Now the SetRange will function according to the case sensitivity of keywords in IndexFieldNames

New Features in IBDAC 8.2

- Added support for RAD Studio 11 Alexandria Release 2
- Added support for Lazarus 2.2.2
- Added support for iOS 15
- Added support for Android 12
- Added the CloneCursor method for Query and Table components that allows sharing data between datasets
- Added support for working with time zone data types when clients don't have the ICU library
- Added support for the isc_dpb_session_time_zone, isc_dpb_set_bind, isc_dpb_decfloat_round, and isc_dpb_decfloat_traps DPB
- Added support for EXTENDED TIME/TIMESTAMP WITH TIME ZONE data types
- Improved support for TIME/TIMESTAMP WITH TIME ZONE data types
- Improved the performance of exporting to XML
- Fixed bug with a "Too many Contexts of Relation/Procedures/Views" error when using "UPDATE OR INSERT" batch operations
- Fixed bug with using CAST in the SQL statement when UseUnicode is set to True
- Fixed bug with reading GUID data of stored procedure in using the AsGuid property
- Fixed bug with processing GUID data in using in parameter

- Fixed bug with working with the transaction when LockMode <> ImNone
- Fixed bug with using the TIBCAlerter component in WebBroker applications
- Fixed bug with "Invalid variant type" error when the DMLRefresh property is set to True
- Fixed bug with reading BLOBs when the CacheBlobs property is set to False and the UseUnicode property is set to True
- Fixed bug when a connection string parameter value contains a single quote

New Features in IBDAC 8.1

- RAD Studio 11 Alexandria Release 1 is supported
- Lazarus 2.2.0 is supported
- Windows 11 is supported
- macOS Monterey is supported

New Features in IBDAC 8.0

- RAD Studio 11 Alexandria is supported
- macOS ARM is supported
- Firebird 4 is supported
- Added demo project for FastReport FMX
- Added the ExplainPlan property that obtains the query execution plan for Table, Query, and SQL components

New Features in IBDAC 7.4

- RAD Studio 10.4.2 Sydney is supported
- macOS 11 Big Sur is supported
- iOS 14 is supported
- Android 11 is supported
- Over-the-Wire (OTW) encryption is supported
- Performance of batch operations is improved
- Memory consumption in batch operations is reduced

- Performance of the FindFirst, FindNext, FindLast, and FindPrior methods is improved
- Automatic detection of computed fields when generating update statements is improved

New Features in IBDAC 7.3

- Lazarus 2.0.10 and FPC 3.2.0 are supported
- Performance of Batch Insert, Update, and Delete operations is improved

New Features in IBDAC 7.2

- RAD Studio 10.4 Sydney is supported
- Lazarus 2.0.8 is supported
- macOS 64-bit in Lazarus is supported

New Features in IBDAC 7.1

- Android 64-bit is supported
- Lazarus 2.0.6 is supported
- Interbase 2020 is supported
- Now Trial edition for macOS and Linux is fully functional
- Improved performance when using pooling
- Handling of BLOB fields in batch operations is improved

New Features in IBDAC 7.0

- macOS 64-bit is supported
- Release 2 for RAD Studio 10.3 Rio, Delphi 10.3 Rio, and C++Builder 10.3 Rio is now required

New Features in IBDAC 6.4

- Lazarus 2.0.2 is supported
- Local connection is supported
- The URL-style connection string format is supported
- The TIBConnectionOptions.IPVersion property is added

- The TCustomIBCSERVICE.IPVersion property is added
- Improved performance when inserting data into a table having BLOB fields using Loader component
- The DefaultSortType property for TVirtualTable is added
- Performance of the SaveToFile/LoadFromFile methods of TVirtualTable is significantly increased

New Features in IBDAC 6.3

- RAD Studio 10.3 Rio is supported
- Possibility to write large blobs by pieces is added
- Support of UPPER and LOWER functions for Unified SQL is added
- The boSkipData option for the BackupService component is added
- The roSkipData option for the RestoreService component is added
- TIBCQuery.OnGetBlobData event is added

New Features in IBDAC 6.2

- Lazarus 1.8.4 is supported
- Support for System Encryption Password (SEP) is added
- Possibility to grant/revoke admin role for a Firebird user with the help of the IBCSecurityService component is added
- Performance of batch operations is improved
- Demo projects for IntraWeb 14 are added
- Now the "Data type is not supported" exception is not raised by the Query component when the DescribeParams property is set to True

New Features in IBDAC 6.1

- Support for Firebird on Android platform is added
- Support for Firebird 3 packages is added
- Aliases handling in the RETURNING clause is supported

- The WireCompression connection parameter for Firebird 3 is supported

New Features in IBDAC 6.0

- RAD Studio 10.2 Tokyo is supported
- Linux in RAD Studio 10.2 Tokyo is supported
- Lazarus 1.6.4 and Free Pascal 3.0.2 is supported
- Possibility to manage batch operations using a transaction is added
- Possibility to obtain active transaction number using DBMonitor is added

New Features in IBDAC 5.7

- RAD Studio 10.1 Berlin is supported
- Lazarus 1.6 and FPC 3.0.0 is supported
- Support for the BETWEEN statement in TDADataset.Filter is added
- Data Type Mapping performance is improved
- Performance of TDALoader on loading data from TDataSet is improved

New Features in IBDAC 5.6

- RAD Studio 10 Seattle is supported
- INSERT, UPDATE and DELETE batch operations are supported
- Now Trial for Win64 is a fully functional Professional Edition

New Features in IBDAC 5.5

- RAD Studio XE8 is supported
- AppMethod is supported
- Firebird 3 is supported
- Firebird 3 BOOLEAN column type is supported
- The roMetadataOnly option in the RestoreService component is added

New Features in IBDAC 5.4

- RAD Studio XE7 is supported

- Lazarus 1.2.4 is supported
- Demo projects for FastReport 5 are added
- The TCustomDADataset.GetKeyFieldNames method is added
- The ConstraintColumns metadata kind for the TDAMetadata component is added

New Features in IBDAC 5.3

- RAD Studio XE6 is supported
- Android in C++Builder XE6 is supported
- Lazarus 1.2.2 and FPC 2.6.4 is supported
- SmartFetch mode for TDataSet descendants is added
- Now update queries inside TDataSet descendants have correct owner
- The TIBCDatasetOptions.MasterFieldsNullable property is added

New Features in IBDAC 5.2

- iOS in C++Builder XE5 is supported
- RAD Studio XE5 Update 2 is now required
- Now .obj and .o files are supplied for C++Builder
- Performance is improved
- Compatibility of migrating floating-point fields from other components is improved

New Features in IBDAC 5.1

- RAD Studio XE5 is supported
- Application development for Android is supported
- Lazarus 1.0.12 is supported
- Performance is improved
- Automatic checking for new versions is added
- Flexible management of conditions in the WHERE clause is added
- The possibility to use conditions is added
- Support of the IN keyword in the TDataSet.Filter property is added

- Like operator behaviour when used in the Filter property is now similar to TClientDataSet
- The possibility to use ranges is added
- The Ping method for the Connection component is added
- The AllowImplicitConnect option for the Connection component is added
- The SQLRecCount property for the Query and StoredProc components is added
- The ScanParams property for the Script component is added
- The RowsAffected property for the Script component is added
- Trusted authentication mode for Firebird is supported
- Migration from FIBPlus is added
- Now the TIBCTransaction.Params property values can be separated by a semicolon
- The ForceUsingDefaultPort global variable is added
- TIBCLoader.LoadFromDataSet is optimized for cases when a dataset record count is less than the RowsPerBatch value

New Features in IBDAC 5.0

- Rad Studio XE4 is supported
- NEXTGEN compiler is supported
- Application development for iOS is supported
- Connection string support is added
- Possibility to encrypt entire tables and datasets is added
- Possibility to determine if data in a field is encrypted is added
- Support of TimeStamp, Single and Extended fields in VirtualTable is added
- InterBase XE3 ToGo Edition support for iOS device is added
- Additional database shutdown options for TIBConfigService.ShutdownDatabase are added

New Features in IBDAC 4.6

- Rad Studio XE3 Update 1 is now required

- C++Builder 64-bit for Windows is supported
- TIBConnection.Port property that allows specifying the port number or the service name for connection is added

New Features in IBDAC 4.5

- Rad Studio XE3 is supported
- Windows 8 is supported

New Features in IBDAC 4.1

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

New Features in InterBase Data Access Components 4.00

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

single batch is added

New Features in InterBase Data Access Components 3.60

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

New Features in InterBase Data Access Components 3.50

- Embarcadero RAD Studio XE supported

New Features in InterBase Data Access Components 3.10

- Embarcadero RAD Studio 2010 supported

New Features in InterBase Data Access Components 3.00

- [TIBCLoader](#) component

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

- **InterBase services components**

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

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

New Features in InterBase Data Access Components

2.70

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

New Features in InterBase Data Access Components

2.50

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

New Features in InterBase Data Access Components

2.20

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

New Features in InterBase Data Access Components

2.10

- C++Builder 2007 supported

New Features in InterBase Data Access Components

2.00

New functionality:

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

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

Support for more InterBase/Firebird server functionality:

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

Extensions and improvements to existing functionality:

- General performance improved
- [Master/detail](#) functionality extensions:
 - [Local master/detail](#) relationship support added
 - Support for master/detail relationships in [CachedUpdates](#) mode added
- [Connection pool](#) functionality improvements:
 - Efficiency significantly improved
 - New [API for draining the connection pool](#) added
- [TIBCScript](#) component improvements:
 - Support for executing [individual statements](#) in scripts added
 - Support for [executing huge scripts stored in files](#) with dynamic loading added
 - Ability to use standard ISQL tool syntax added
- Greatly increased [performance of applying updates](#) in [CachedUpdates](#) mode

- Working with [calculated and lookup fields improvements](#):
 - Local [sorting](#) and filtering added
 - Record [location](#) speed increased
 - Improved working with lookup fields
- Ability to customize update commands by attaching external components to [TIBCUpdateSQL](#) objects added
- Ability to [include all fields](#) in automatically generated update SQLs added

Usability improvements:

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

New Features in InterBase Data Access Components 1.10

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

2 General Information

This section contains general information about InterBase Data Access Components

- [Overview](#)
- [Features](#)
- [Requirements](#)
- [Compatibility](#)
- [Using Several DAC Products in One IDE](#)
- [Component List](#)
- [Hierarchy Chart](#)
- [Editions](#)
- [Licensing and Subscriptions](#)

- [Getting Support](#)
- [Frequently Asked Questions](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

2.1 Overview

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

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

Advantages of IBDAC Technology

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

Wide Coverage of InterBase Features

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

Optimized Code

The goal of IBDAC is to enable developers to write efficient and flexible database applications. The IBDAC library is implemented using optimized code and advanced data access algorithms. Component interfaces undergo comprehensive performance tests and are designed to help you write thin and efficient product data access layers. Find out more about

how to use IBDAC to optimize your database applications in [Increasing Performance](#).

Compatibility with other Connectivity Methods

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

Development and Support

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

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

Key Features

- Direct access to server data. Does not require installation of other data provider layers (such as BDE and ODBC)
- VCL, LCL and FireMonkey versions of the library available
- Full support of the latest versions of [InterBase and Firebird database servers](#)
- Support for all InterBase data types
- [Disconnected Model](#) with automatic connection control for working with data offline
- [Local Failover](#) for detecting connection loss and implicitly re-executing certain operations
- All types of local [sorting](#) and [filtering](#), including by calculated and lookup fields
- Automatic [data updating](#) with [TIBCQuery](#) and [TIBCTable](#) components
- [Unicode and national charsets](#) support
- InterBase [Events](#) support
- InterBase [OTW encryption](#) support
- Advanced script execution functionality with [TIBCScript](#) component

- [Support for using Macros in SQL](#)
- Easy migration from BDE and IBX with [Migration Wizard](#)
- Lets you use Professional Edition of [Delphi and C++Builder](#) to develop client/server applications
- Included annual [IBDAC Subscription](#) with [Priority Support](#)
- Licensed royalty-free per developer, per team, or per site

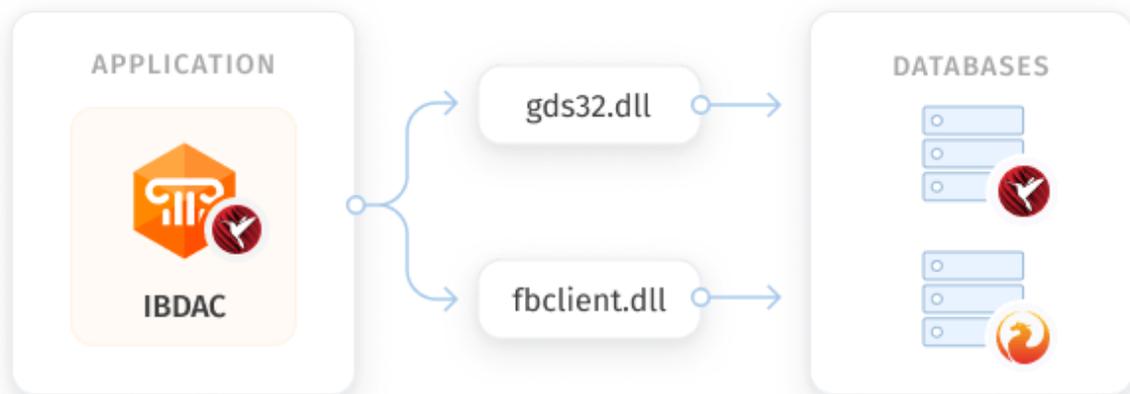
The full list of IBDAC features can be found in [Features](#).

How does IBDAC work?

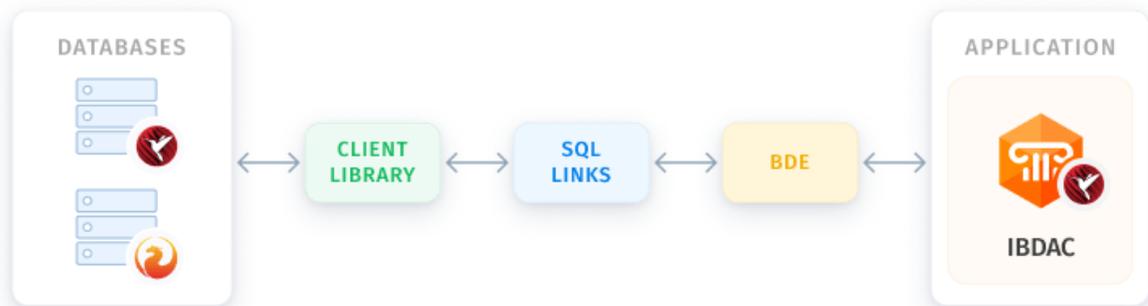
IBDAC uses InterBase client software to connect to the server directly through the native InterBase interface, without using BDE or ODBC. It is designed to be lightweight. It consists of a minimal layer between InterBase server and your code. This extends functionality without sacrificing performance.

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

IBDAC Connection



BDE Connection



© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

2.2 Features

Supported target platforms

- Windows 32-bit and 64-bit
- macOS 64-bit
- Mac ARM
- iOS 64-bit
- iOS Simulator ARM 64-bit
- Android 32-bit and 64-bit
- Linux 32-bit (only in Lazarus and Free Pascal) and 64-bit

General usability:

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

Network and connectivity:

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

Compatibility:

- [Full support of the latest versions of InterBase and Firebird database servers](#)
- Support for InterBase ToGo Edition
- Support for all InterBase and Firebird data types
- [Compatible with all IDE versions starting with Delphi 6, C++Builder 6 and Lazarus \(Free Pascal\)](#)
- Includes provider for UniDAC Express Edition
- [Wide reporting component support](#), including support for InfoPower, ReportBuilder, FastReport
- Support of all standard and third-party visual data-aware controls
- Allows you to use Professional Edition of Delphi and C++Builder to develop client/server applications

InterBase technology support:

- Support for fast record insertion with the [TIBCLoader](#) component
- [InterBase event](#) support
- Comprehensive [array data type](#) support
- [Advanced BLOB](#) support
- [Streaming](#) (non-caching) BLOB access support
- [Advanced generator](#) support
- Advanced support for the character set OCTETS
- Support for the Firebird 2 EXECUTE BLOCK syntax
- Support for the Firebird 2 [RETURNING clause](#)
- [Advanced locking](#) for Firebird 2

- [Automatic updates](#) by [DB_KEY](#) unique field for Firebird 2
- [Default value support for stored procedures](#)
- InterBase services components for configuring server parameters and security
- Support for the Firebird 3 BOOLEAN datatype
- Support for the Firebird 2.1 trusted authentication

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](#)
- Deferred BLOB and array fields reading

Local data storage operations:

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

Data access and data management automation:

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

Extended data access functionality:

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

Data exchange:

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

Script execution:

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

SQL execution monitoring:

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

Visual extensions:

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

Design-time enhancements:

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

Resources:

- Code documentation and guides in the CHM, PDF, and HXS formats
- Many helpful [demo](#) projects

Licensing and support:

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

2.3 Requirements

Applications based on IBDAC require client software only (the gds32.dll or fbclient.dll libraries for InterBase or Firebird correspondingly). IBDAC dynamically loads the client library available on user systems. By default, IBDAC searches client libraries in directories specified in the PATH environment variable. The [TIBConnection.ClientLibrary](#) property is used to specify the path to the client library. Starting with Delphi XE2 it is possible to develop 64-bit applications. To develop 64-bit applications, you should use 64-bit client libraries. For more information, please refer to the ["Deployment"](#) and ["Database Specific Aspects of 64-bit Development"](#) articles.

See Also

- [FAQ: What software should be installed on a client computer so that my applications that use IBDAC can run?](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

2.4 Compatibility

Database Server Compatibility

InterBase	Windows	macOS	Linux	iOS	Android
Versions since XE3 up to 2020	✓	✓	✓	✓	✓
Versions since XE	✓	✓	✓		
Versions since 4.2	✓		✓		
Firebird					
Versions: 1.x, 2.x, 3.x, 4.x	✓	✓	✓		✓

IDE Compatibility

IBDAC is compatible with the following IDEs:

Embarcadero RAD Studio 12 Athens

- Embarcadero Delphi 12 Athens for Windows
- Embarcadero Delphi 12 Athens for macOS
- Embarcadero Delphi 12 Athens for Linux
- Embarcadero Delphi 12 Athens for iOS

- Embarcadero Delphi 12 Athens for Android
- Embarcadero C++Builder 12 Athens for Windows
- Embarcadero C++Builder 12 Athens for iOS
- Embarcadero C++Builder 12 Athens for Android

Embarcadero RAD Studio 11.1 Alexandria

- Embarcadero Delphi 11.1 Alexandria for Windows
- Embarcadero Delphi 11.1 Alexandria for macOS
- Embarcadero Delphi 11.1 Alexandria for Linux
- Embarcadero Delphi 11.1 Alexandria for iOS
- Embarcadero Delphi 11.1 Alexandria for Android
- Embarcadero C++Builder 11.1 Alexandria for Windows
- Embarcadero C++Builder 11.1 Alexandria for iOS
- Embarcadero C++Builder 11.1 Alexandria for Android

Embarcadero RAD Studio 10.4 Sydney (Requires Release 1 or Release 2)

- Embarcadero Delphi 10.4 Sydney for Windows
- Embarcadero Delphi 10.4 Sydney for macOS
- Embarcadero Delphi 10.4 Sydney for Linux
- Embarcadero Delphi 10.4 Sydney for iOS
- Embarcadero Delphi 10.4 Sydney for Android
- Embarcadero C++Builder 10.4 Sydney for Windows
- Embarcadero C++Builder 10.4 Sydney for iOS
- Embarcadero C++Builder 10.4 Sydney for Android

Embarcadero RAD Studio 10.3 Rio (Requires [Release 2](#) or [Release 3](#))

- Embarcadero Delphi 10.3 Rio for Windows
- Embarcadero Delphi 10.3 Rio for macOS
- Embarcadero Delphi 10.3 Rio for Linux
- Embarcadero Delphi 10.3 Rio for iOS
- Embarcadero Delphi 10.3 Rio for Android

- Embarcadero C++Builder 10.3 Rio for Windows
- Embarcadero C++Builder 10.3 Rio for macOS
- Embarcadero C++Builder 10.3 Rio for iOS
- Embarcadero C++Builder 10.3 Rio for Android

Embarcadero RAD Studio 10.2 Tokyo (Incompatible with Release 1)

- Embarcadero Delphi 10.2 Tokyo for Windows
- Embarcadero Delphi 10.2 Tokyo for macOS
- Embarcadero Delphi 10.2 Tokyo for Linux
- Embarcadero Delphi 10.2 Tokyo for iOS
- Embarcadero Delphi 10.2 Tokyo for Android
- Embarcadero C++Builder 10.2 Tokyo for Windows
- Embarcadero C++Builder 10.2 Tokyo for macOS
- Embarcadero C++Builder 10.2 Tokyo for iOS
- Embarcadero C++Builder 10.2 Tokyo for Android

Embarcadero RAD Studio 10.1 Berlin

- Embarcadero Delphi 10.1 Berlin for Windows
- Embarcadero Delphi 10.1 Berlin for macOS
- Embarcadero Delphi 10.1 Berlin for iOS
- Embarcadero Delphi 10.1 Berlin for Android
- Embarcadero C++Builder 10.1 Berlin for Windows
- Embarcadero C++Builder 10.1 Berlin for macOS
- Embarcadero C++Builder 10.1 Berlin for iOS
- Embarcadero C++Builder 10.1 Berlin for Android

Embarcadero RAD Studio 10 Seattle

- Embarcadero Delphi 10 Seattle for Windows
- Embarcadero Delphi 10 Seattle for macOS
- Embarcadero Delphi 10 Seattle for iOS
- Embarcadero Delphi 10 Seattle for Android

- Embarcadero C++Builder 10 Seattle for Windows
- Embarcadero C++Builder 10 Seattle for macOS
- Embarcadero C++Builder 10 Seattle for iOS
- Embarcadero C++Builder 10 Seattle for Android

Embarcadero RAD Studio XE8

- Embarcadero Delphi XE8 for Windows
- Embarcadero Delphi XE8 for macOS
- Embarcadero Delphi XE8 for iOS
- Embarcadero Delphi XE8 for Android
- Embarcadero C++Builder XE8 for Windows
- Embarcadero C++Builder XE8 for macOS
- Embarcadero C++Builder XE8 for iOS
- Embarcadero C++Builder XE8 for Android

Embarcadero RAD Studio XE7

- Embarcadero Delphi XE7 for Windows
- Embarcadero Delphi XE7 for macOS
- Embarcadero Delphi XE7 for iOS
- Embarcadero Delphi XE7 for Android
- Embarcadero C++Builder XE7 for Windows
- Embarcadero C++Builder XE7 for macOS
- Embarcadero C++Builder XE7 for iOS
- Embarcadero C++Builder XE7 for Android

Embarcadero RAD Studio XE6

- Embarcadero Delphi XE6 for Windows
- Embarcadero Delphi XE6 for macOS
- Embarcadero Delphi XE6 for iOS
- Embarcadero Delphi XE6 for Android
- Embarcadero C++Builder XE6 for Windows

- Embarcadero C++Builder XE6 for macOS
- Embarcadero C++Builder XE6 for iOS
- Embarcadero C++Builder XE6 for Android

Embarcadero RAD Studio XE5 (Requires [Update 2](#))

- Embarcadero Delphi XE5 for Windows
- Embarcadero Delphi XE5 for macOS
- Embarcadero Delphi XE5 for iOS
- Embarcadero Delphi XE5 for Android
- Embarcadero C++Builder XE5 for Windows
- Embarcadero C++Builder XE5 for macOS
- Embarcadero C++Builder XE5 for iOS

Embarcadero RAD Studio XE4

- Embarcadero Delphi XE4 for Windows
- Embarcadero Delphi XE4 for macOS
- Embarcadero Delphi XE4 for iOS
- Embarcadero C++Builder XE4 for Windows
- Embarcadero C++Builder XE4 for macOS

Embarcadero RAD Studio XE3 (Requires [Update 2](#))

- Embarcadero Delphi XE3 for Windows
- Embarcadero Delphi XE3 for macOS
- Embarcadero C++Builder XE3 for Windows
- Embarcadero C++Builder XE3 for macOS

Embarcadero RAD Studio XE2 (Requires [Update 4 Hotfix 1](#))

- Embarcadero Delphi XE2 for Windows
- Embarcadero Delphi XE2 for macOS
- Embarcadero C++Builder XE2 for Windows
- Embarcadero C++Builder XE2 for macOS

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
- CodeGear C++Builder 2007

Borland Developer Studio 2006

- Borland Delphi 2006
- Borland C++Builder 2006

Borland Delphi 7

Borland Delphi 6 (Requires [Update Pack 2](#) – Delphi 6 Build 6.240)

Borland C++Builder 6 (Requires [Update Pack 4](#) – C++Builder 6 Build 10.166)

[Lazarus 3.2.0](#) and [Free Pascal 3.2.2](#) for Windows, macOS, and Linux (32-bit and 64-bit)

All the existing Delphi and C++Builder editions are supported: Architect, Enterprise, Professional, Community, and Starter.

Lazarus and Free Pascal are supported only in Trial Edition and Professional Edition with source code.

Supported Target Platforms

- Windows 32-bit and 64-bit
- macOS 64-bit and ARM (Apple Silicon M1)
- Linux 32-bit (only in Lazarus and Free Pascal) and 64-bit
- iOS 64-bit
- iOS Simulator ARM 64-bit
- Android 32-bit and 64-bit

Support for Windows 64-bit is available since RAD Studio XE2. Support for iOS 64-bit is

available since RAD Studio XE8. Support for Android 32-bit is available since RAD Studio XE5. Support for Linux 64-bit is available since RAD Studio 10.2 Tokyo. Support for macOS 64-bit is available since RAD Studio 10.3 Rio. Support for Android 64-bit is available since RAD Studio 10.3.3 Rio.

Supported GUI Frameworks

- FireMonkey (FMX)
- Visual Component Library (VCL)
- Lazarus Component Library (LCL)

Devart Data Access Components Compatibility

All DAC products are compatible with each other.

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

2.5 Using Several DAC Products in One IDE

UniDAC, ODAC, SDAC, MyDAC, IBDAC, PgDAC, LiteDAC and VirtualDAC components use common base packages listed below:

Packages:

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

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

2.6 Component List

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

Basic IBDAC components

	TIBCConnection	Sets and controls connection to InterBase database.
	TIBCTransaction	Provides discrete transaction control over database connections.
	TIBCQuery	Uses SQL statements to retrieve data from InterBase table or tables. Single SELECT statement may be adequately used to generate missing INSERT, DELETE, UPDATE statements.
	TIBCSQL	Executes SQL statements and stored procedures, which do not return rowsets.
	TIBCTable	Lets you retrieve and update data in a single table without writing SQL statements.
	TIBCStoredProc	Executes stored procedures and functions, allows to edit cursor data returned as parameter.
	TIBCUpdateSQL	Lets you tune update operations for a DataSet component.
	TIBCDataSource	Provides an interface between an IBDAC dataset components and data-aware controls on a form.
	TIBCScript	Executes sequences of SQL statements.
	TIBCSQLMonitor	Use to monitor dynamic SQL execution in IBDAC based applications.
	TIBCConnectDialog	Used to build custom prompts for username, password and server name.

	TVirtualTable	Dataset that stores data in memory. This component is placed on the Data Access page of the Component palette.
	TVirtualDataSet	Dataset that processes arbitrary non-tabular data.

IBDAC Professional Edition components

	TIBCEncryptor	Represents data encryption and decryption in client application.
	TIBCLoader	Allows to load external data into the database table.
	TIBCAlerter	Use to transfer messages between connections.
	TIBCMetaData	Retrieves metadata on specified SQL object.
	TIBCServerProperties	Returns database server information, including configuration parameters, and also version and license information.
	TIBCConfigService	Configures database parameters.
	TIBCLicensingService	Adds or removes InterBase software activation certificates.
	TIBCLogService	Returns the contents of the interbase.log file from server.
	TIBCStatisticalService	Shows database statistics.
	TIBCValidationService	Validates a database and reconciles database transactions.
	TIBCSecurityService	Used to manage user access to the InterBase server.
	TIBCTraceService	Used for working with trace service added in Firebird 2.5.
	TIBCBackupService	Used to backup a database.
	TIBCRestoreService	Used to restore a database.

	TCRBatchMove	Transfers data between all types of TDataSet descendants. This component is placed on the Data Access page of the Component palette.
---	------------------------------	--

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

2.7 Hierarchy Chart

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

TObject

```

|-TPersistent
|
|   |-TComponent
|   |
|   |   |-TCustomConnection
|   |   |
|   |   |   |-TCustomDAConnection
|   |   |   |
|   |   |   |   |-TIBCConnection
|   |   |   |
|   |   |   |-TDataSet
|   |   |   |
|   |   |   |   |-TMemDataSet
|   |   |   |   |
|   |   |   |   |   |-TCustomDADataSet
|   |   |   |   |   |
|   |   |   |   |   |   |-TCustomIBCDataSet
|   |   |   |   |   |   |
|   |   |   |   |   |   |   |-TCustomIBCQuery
|   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |-TIBCQuery
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |-TIBCStoredProc
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |-TCustomIBCTable
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |-TIBCTable
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |-TDAMetaData
|   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |-TIBCMetaData
|   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |-TVirtualTable
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |-TDataSource
|   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |-TCRDataSource
|   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |-TIBCDataSource

```

- | [|-DADDataAdapter](#)
- | | [|-IBCDataAdapter](#)
- | [|-TCRBatchMove](#)
- | [|-TCustomConnectDialog](#)
- | | [|-TIBCCConnectDialog](#)
- | [|-TCustomDASQL](#)
- | | [|-TIBCSQL](#)
- | [|-TCustomDASQLMonitor](#)
- | | [|-TIBCSQLMonitor](#)
- | [|-TCustomDAUpdateSQL](#)
- | | [|-TIBCUpdateSQL](#)
- | [|-TDALoader](#)
- | | [|-TIBCLoader](#)
- | [|-TDAScript](#)
- | | [|-TIBCScript](#)
- | [|-TDAAlerter](#)
- | | [|-TIBCAlerter](#)
- | [|-TCREncryptor](#)
- | | [|-TIBCEncryptor](#)
- | [|-TDATransaction](#)
- | | [|-TIBCTransaction](#)
- | [|-TCustomIBCService](#)
 - | [|-TIBCServerProperties](#)
 - | [|-TIBCConfigService](#)
 - | [|-TIBCLicensingService](#)
 - | [|-TIBCControlAndQueryService](#)
 - | [|-TIBCLogService](#)
 - | [|-TIBCStatisticalService](#)
 - | [|-TIBCValidationService](#)
 - | [|-TIBCSecurityService](#)
 - | [|-TIBCTraceService](#)
 - | [|-TIBCBackupRestoreService](#)

Feature	Standard	Professional
Desktop Application Development		
Windows	✓	✓
macOS	✗	✓
Linux	✗	✓
Mobile Application Development		
iOS	✗	✓
Android	✗	✓
Data Access Components		
Base Components: TIBCConnection TIBCQuery TIBCSQL TIBCTable TIBCStoredProc TIBCUpdateSQL TIBCDatasource	✓	✓
Script Executing TIBCScript	✓	✓
Transactions managing TIBCTransaction	✓	✓
Fast data loading into the server TIBCLoader	✗	✓
Interbase & Firebird Specific Components		
Messaging between sessions and applications TIBCAlerter	✓	✓
Obtaining metadata about database objects TIBCMetaData	✓	✓
Database server information return TIBCServerProperties	✗	✓
InterBase Services		
Database parameters configurations TIBCConfigService	✗	✓

Managing of activation certificates TIBCLicensingService	×	✓
The server log return TIBCLogService	×	✓
Shows database statistics TIBCStatisticalService	×	✓
Database backup TIBCBackupService	×	✓
Database restore TIBCRestoreService	×	✓
Validation of database TIBCValidationService	×	✓
User access management TIBCSecurityService	×	✓
Interacting with Firebird 2.5 trace service TIBCTraceService	×	✓
DataBase Activity Monitoring		
Monitoring of per-component SQL execution TIBCSQLMonitor	✓	✓
Additional Components		
Advanced connection dialog TIBCConnectDialog	✓	✓
Data encryption and decryption TIBCEncryptor	×	✓
Data storing in memory table TVirtualTable	✓	✓
Dataset that wraps arbitrary non-tabular data TVirtualDataSet	✓	✓
Advanced DBGrid with extended functionality TCRDBGrid	✓	✓
Records transferring between datasets TCRBatchMove	×	✓
Design-Time Features		
Enhanced component and property editors	✓	✓
Migration Wizard	✓	✓
DataSet Manager	×	✓

Cross IDE Support

Lazarus and Free Pascal Support



SRC¹

¹ Available only in Professional Edition with Source Code.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

2.9 Licensing

PLEASE READ THIS LICENSE AGREEMENT CAREFULLY. BY INSTALLING OR USING THIS SOFTWARE, YOU INDICATE ACCEPTANCE OF AND AGREE TO BECOME BOUND BY THE TERMS AND CONDITIONS OF THIS LICENSE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, DO NOT INSTALL OR USE THIS SOFTWARE AND PROMPTLY RETURN IT TO DEVART.

INTRODUCTION

This Devart end-user license agreement ("Agreement") is a legal agreement between you (either an individual person or a single legal entity) and Devart, for the use of IBDAC software application, source code, demos, intermediate files, printed materials, and online or electronic documentation contained in this installation file. For the purpose of this Agreement, the software program(s) and supporting documentation will be referred to as the "Software".

LICENSE

1. GRANT OF LICENSE

The enclosed Software is licensed, not sold. You have the following rights and privileges, subject to all limitations, restrictions, and policies specified in this Agreement.

1.1. If you are a legally licensed user, depending on the license type specified in the registration letter you have received from Devart upon purchase of the Software, you are entitled to either:

- install and use the Software on one or more computers, provided it is used by 1 (one) for the sole purposes of developing, testing, and deploying applications in accordance with this

Agreement (the "Single Developer License"); or - install and use the Software on one or more computers, provided it is used by up to 4 (four) developers within a single company at one physical address for the sole purposes of developing, testing, and deploying applications in accordance with this Agreement (the "Team Developer License"); or - install and use the Software on one or more computers, provided it is used by developers in a single company at one physical address for the sole purposes of developing, testing, and deploying applications in accordance with this Agreement (the "Site License").

1.2. If you are a legally licensed user of the Software, you are also entitled to:

- make one copy of the Software for archival purposes only, or copy the Software onto the hard disk of your computer and retain the original for archival purposes;
- develop and test applications with the Software, subject to the Limitations below;
- create libraries, components, and frameworks derived from the Software for personal use only;
- deploy and register run-time libraries and packages of the Software, subject to the Redistribution policy defined below.

1.3. You are allowed to use evaluation versions of the Software as specified in the Evaluation section.

No other rights or privileges are granted in this Agreement.

2. LIMITATIONS

Only legally registered users are licensed to use the Software, subject to all of the conditions of this Agreement. Usage of the Software is subject to the following restrictions.

2.1. You may not reverse engineer, decompile, or disassemble the Software.

2.2. You may not build any other components through inheritance for public distribution or commercial sale.

2.3. You may not use any part of the source code of the Software (original or modified) to build any other components for public distribution or commercial sale.

2.4. You may not reproduce or distribute any Software documentation without express written permission from Devart.

2.5. You may not distribute and sell any portion of the Software without integrating it into your Applications as Executable Code, except a Trial version that can be distributed for free as original Devart's IBDAC Trial package.

2.6. You may not transfer, assign, or modify the Software in whole or in part. In particular, the Software license is non-transferable, and you may not transfer the Software installation package.

2.7. You may not remove or alter any Devart's copyright, trademark, or other proprietary rights notice contained in any portion of Devart units, source code, or other files that bear such a notice.

3. REDISTRIBUTION

The license grants you a non-exclusive right to compile, reproduce, and distribute any new software programs created using IBDAC. You can distribute IBDAC only in compiled Executable Programs or Dynamic-Link Libraries with required run-time libraries and packages.

All Devart's units, source code, and other files remain Devart's exclusive property.

4. TRANSFER

You may not transfer the Software to any individual or entity without express written permission from Devart. In particular, you may not share copies of the Software under "Single Developer License" and "Team License" with other co-developers without obtaining proper license of these copies for each individual.

5. TERMINATION

Devart may immediately terminate this Agreement without notice or judicial resolution in the event of any failure to comply with any provision of this Agreement. Upon such termination you must destroy the Software, all accompanying written materials, and all copies.

6. EVALUATION

Devart may provide evaluation ("Trial") versions of the Software. You may transfer or distribute Trial versions of the Software as an original installation package only. If the Software you have obtained is marked as a "Trial" version, you may install and use the Software for a period of up to 60 calendar days from the date of installation (the "Trial Period"), subject to the additional restriction that it is used solely for evaluation of the Software and not in conjunction with the development or deployment of any application in production. You may not use applications developed using Trial versions of the Software for any commercial purposes. Upon expiration of the Trial Period, the Software must be uninstalled, all its copies and all

accompanying written materials must be destroyed.

7. WARRANTY

The Software and documentation are provided "AS IS" without warranty of any kind. Devart makes no warranties, expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or use.

8. SUBSCRIPTION AND SUPPORT

The Software is sold on a subscription basis. The Software subscription entitles you to download improvements and enhancement from Devart's web site as they become available, during the active subscription period. The initial subscription period is one year from the date of purchase of the license. The subscription is automatically activated upon purchase, and may be subsequently renewed by Devart, subject to receipt applicable fees. Licensed users of the Software with an active subscription may request technical assistance with using the Software over email from the Software development. Devart shall use its reasonable endeavours to answer queries raised, but does not guarantee that your queries or problems will be fixed or solved.

Devart reserves the right to cease offering and providing support for legacy IDE versions.

9. COPYRIGHT

The Software is confidential and proprietary copyrighted work of Devart and is protected by international copyright laws and treaty provisions. You may not remove the copyright notice from any copy of the Software or any copy of the written materials, accompanying the Software.

This Agreement contains the total agreement between the two parties and supersedes any other agreements, written, oral, expressed, or implied.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

2.10 Getting Support

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

Support Options

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

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

If you have a question about ordering IBDAC or any other Devart product, please contact sales@devart.com.

IBDAC Priority Support

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

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

- The version of Delphi, C++Builder you are using.
- Your IBDAC Registration number.
- Full IBDAC edition name and version number. You can find both of these from the IBDAC | IBDAC About menu in the IDE.
- Versions of the InterBase server and client you are using.
- A detailed problem description.
- If possible, a small test project that reproduces the problem. It is recommended to use Scott or SYS schema objects only. Please include definitions for all and avoid using third-party components.

Devart. All Rights Reserved.

2.11 Frequently Asked Questions

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

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

Installation and Deployment

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

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

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

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

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

The minimal configuration of client installation includes the following steps:

- Copy the client file *gds32.dll* to the folder available for executable unit of your program. For example, to the folder with your executable file, or to the Windows system folder. For more information, see description of the *LoadLibrary* function and the environment variable *PATH*.
- Add the "gds_db 3050/tcp" line to the *services* file in the *%WinDir%\system32\drivers\etc* directory.

For Firebird version 1.0.0.338 and higher, both client and server use port 3050 by default. So, you do not need to modify the *services* file. You can also specify port number for the Firebird client in connection string - *server/3050:c:\dir\data.gdb*

- Copy file *InterBase.msg* (or *firebird.msg* for Firebird) to the folder available for executable unit of your program. File must belong to the same version as InterBase or Firebird.

Licensing and Subscriptions

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

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

2. Can I create components using IBDAC?

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

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

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

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

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

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

Edition Correspondence Table for Upgrading to IBDAC 2.00

Old Edition Level	New Edition Level
IBDAC Standard Edition	IBDAC Professional Edition
IBDAC Professional	IBDAC Professional

Edition	Edition with Sources
IBDAC Trial Edition	IBDAC Trial Edition

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

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

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

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

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

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

Users with a valid IBDAC Subscription get the following benefits:

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

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

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

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

Yes, you can. IBDAC version licenses are perpetual.

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

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

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

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

Yes.

10. How do I upgrade to IBDAC 2.00?

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

Performance

1. How productive is IBDAC?

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

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

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

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

How To

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

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

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

You can determine your IBDAC version number in several ways:

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

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

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

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

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

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

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

- Fill the `SQLInsert` property with the parametrized 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 IBDAC to perform the last step.

Setting these properties is optional and allows you to automatically execute additional SQL

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

5. Some questions about the visual part of IBDAC

The following questions usually arise from the same problem:

- 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 LoginPromp set to True, the connect dialog does not appear.

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

General Questions

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

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

IBDAC is a specialized set of components to access InterBase server with advanced design-time and component interface similar to BDE.

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

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

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

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

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

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

3. Are the IBDAC connection components thread-safe?

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

4. Behaviour of my application has changed when I upgraded IBDAC. How can I restore the old behaviour with the new version?

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

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

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

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

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

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

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

3 Getting Started

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

- [What is IBDAC?](#)
- [Installing IBDAC.](#)
- [Working with the IBDAC demo projects.](#)
- [Compiling and deploying your IBDAC project.](#)

- [Using the IBDAC documentation.](#)
- [How to get help with IBDAC.](#)

What is IBDAC?

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

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

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

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

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

Installing IBDAC

To install IBDAC, complete the following steps.

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

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

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

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

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

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

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

to the Menu bar.

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

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

Working with the IBDAC demo projects

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

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

IbDacDemo Walkthrough

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

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

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

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

At this point, you will be able to browse through the available demos, read their descriptions, view their source code, and see the functionality provided by each demo for interacting with InterBase. However, you will not be able to actually retrieve data from InterBase or execute commands until you connect to the database.

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

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

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

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

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

"An error has occurred:

unsuccessful metadata update Table DEPT already exists

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

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

Ignore this exception?"

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

7. Choose a demo that demonstrates an aspect of working with InterBase that you are interested in, and play with the demo frame in the view window on the right. For example, to find out more about how to work with InterBase tables, select the Table demo from the

"Working with Components" folder. A simple InterBase table browser will open in the view panel which will let you open a table in your database by specifying its name and clicking on the Open button.

8. Click on the "Demo source" button in the *IbDacDemo* toolbar to find out how the demo you have selected was implemented. The source code behind the demo project will appear in the view panel. Try to find the places where IBDAC components are used to connect to the database.
9. Click on the "Form as text" button in the *IbDacDemo* toolbar to view the code behind the interface to the demo. Try to find the places where IBDAC components are created on the demo form.
10. Repeat these steps for other demos listed in the explorer window. The available demos are organized in three folders.

Working with components

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

General demos

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

InterBase-specific demos

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

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

Other IBDAC demo projects

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

Compiling and deploying your IBDAC project

Compiling IBDAC-based projects

By default, to compile a project that uses IBDAC classes, your IDE compiler needs to have access to the IBDAC dcu (obj) files. If you are compiling with runtime packages, the compiler will also need to have access to the IBDAC bpl files. **All the appropriate settings for both of these scenarios should take place automatically during the installation of IBDAC.**

You should only need to modify your environment manually if you are using one of the IBDAC editions that comes with source code - IBDAC Professional Edition with Source Code.

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

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

Deploying IBDAC-based projects

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

- The InterBase Client software, if connecting in Client mode.
- The IBDAC bpl files, if compiling with runtime packages.

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

A list of the files which may need to be deployed with IBDAC-based applications is included in the [Deployment](#) topic.

Using the IBDAC documentation

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

The IBDAC documentation includes a detailed reference of all IBDAC components and classes. Many of the IBDAC components and classes inherit or implement members from other VCL, LCL and FireMonkey classes and interfaces. The product documentation also

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

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

How to get help with IBDAC

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

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

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

3.1 Installation

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

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

Before installing IBDAC ...

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

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

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

Installed packages

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

Delphi/C++Builder Win32 project packages

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

Additional packages for using IBDAC managers and wizards

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

datasetmanagerXX.bpl	DataSet Manager package	Delphi\Bin
oramigwizardXX.dll	IBDAC BDE\IBX Migration wizard	%IBDAC%\Bin

Environment Changes

To compile IBDAC-based applications, your environment must be configured to have access to the IBDAC libraries. Environment changes are IDE-dependent.

For all instructions, replace %IBDAC% with the path to your IBDAC installation directory

Delphi

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

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

C++Builder

C++Builder 6:

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

C++Builder 2006, 2007:

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

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

Lazarus

The IBDAC installation program only copies IBDAC files. You need to install IBDAC packages to Lazarus IDE manually. Open %IBDAC%\Source\Lazarus1\dclibdac10.lpk (for Trial version %IBDAC%\Packages\dclibdac10.lpk) file in Lazarus and press the Install button. After that Lazarus IDE will be rebuilt with IBDAC packages.

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

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

Installation of Additional Components and Add-ins

DBMonitor

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

3.2 Migrating from BDE and IBX

Note: Migration Wizard is only available for Delphi.

Migration Wizard allows you to convert your BDE or IBX projects to IBDAC. This wizard replaces BDE or IBX components in a specified project (.dfm and .pas files) with IBDAC components.

To convert a project, perform the following steps.

- Select **Migration Wizard** from the **IBDAC** menu
- Select **Replace BDE components** or **Replace IBX components** to replace the corresponding components with IBDAC ones and click the Next button.
- Select the location of the files to search - current open project or disc folder.

- If you have selected Disc folder on the previous step, specify the required folder and specify whether to process subfolders. Press the Next button.
- Select whether to make backup (it is highly recommended to make a backup), backup location, and log parameters, and press the Next button. Default backup location is RBackup folder in your project folder.
- Check your settings and press the Finish button to start the conversion operation.
- The project should be saved before conversion. You will be asked before saving it. Click Yes to continue project conversion.

After the project conversion it will be reopened.

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

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

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

Your project will be restored to its previous state.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

3.3 Connecting to InterBase and Firebird

This tutorial describes how to connect to InterBase and Firebird using the [TIBConnection](#) component.

1. [Requirements](#)
2. [General Information](#)

3. [Creating a Connection](#)
 - 3.1 [Connecting at Design-Time](#)
 - 3.1.1 [Using TIBConnection Editor](#)
 - 3.1.2 [Using Object Inspector](#)
 - 3.2 [Connecting at Runtime](#)
4. [Opening a Connection](#)
5. [Modifying a Connection](#)
6. [Closing a Connection](#)
7. [Additional Information](#)
8. [See Also](#)

Requirements

This tutorial assumes that you have installed IBDAC and run the database server and the IDE. You need to know the server address, the port number (if you use a port other than the default port 3050), the path to the database file (.gdb or .fdb), and the username and password. To connect at runtime, add the [IBC](#) unit to the `uses` clause for Delphi or include the `IBC.hpp` header file for C++ Builder.

General Information

To establish a connection to the server, set up the properties of the [TIBConnection](#) component: [Server](#), [Port](#), [Database](#), [ClientLibrary](#), [Username](#), and [Password](#). You can also specify all connection parameters in the [ConnectionString](#) property.

Creating a Connection

Connecting at Design-Time

The following assumes that you have already created or opened an existing form in the IDE. At design-time, you can set up a `TIBConnection` object in the TIBConnection Editor or Object Inspector.

1. Find the `TIBConnection` component in the `IBDAC` category on the Tool Palette.
2. Double-click the component. A new object will appear on the form. If this is the first

TIBCCONNECTION object in this unit, it will be named IBCCONNECTION1.

Using TIBCCONNECTION Editor

1. Double-click the IBCCONNECTION1 object.
2. Specify the DNS name or IP address of the InterBase or Firebird server in the SERVER edit box.
3. If you use a port other than the default port 3050, specify it in the PORT edit box.
4. Specify the database file path in the DATABASE edit box, e.g., D:\InterBase\employee.gdb or D:\Firebird\employee.fdb.
5. Specify the username (SYSDBA by default) in the USERNAME edit box.
6. Specify the password (MASTERKEY by default) in the PASSWORD edit box.
7. If you have both InterBase and Firebird client libraries installed, specify the path to the client library — GDS32.DLL for InterBase or FBCLIENT.DLL for Firebird — in the CLIENT LIBRARY edit box. Otherwise, skip this step.

Using Object Inspector

1. Select the IBCCONNECTION1 object on the form.
2. If you have both InterBase and Firebird client libraries installed, set the CLIENTLIBRARY property to GDS32.DLL for InterBase or FBCLIENT.DLL for Firebird. Otherwise, skip this step.
3. Set the DATABASE property to the database file path, e.g., D:\InterBase\employee.gdb or D:\Firebird\employee.fdb.
4. Set the PASSWORD property to the password (MASTERKEY by default).
5. If you use a port other than the default port 3050, set the PORT property to your port.
6. Set the SERVER property to the DNS name or IP address of the InterBase or Firebird server.
7. Set the USERNAME property to the username (SYSDBA by default).

Connecting at Runtime

The same connection parameters at runtime are set up as follows:

Delphi

```
var  
  IBCCONNECTION1: TIBCCONNECTION;  
begin
```

```

IBConnection1 := TIBConnection.Create(nil);
try
  // adds connection parameters
  // if Server is empty, a connection is established through the local protocol
  IBConnection1.Server := 'server';
  IBConnection1.Database := 'database';
  IBConnection1.Username := 'username';
  IBConnection1.Password := 'password';
  IBConnection1.Port := 3050;
  // indicates the client lib for InterBase (for Firebird, use fbclient.dll)
  IBConnection1.ClientLibrary := 'gds32.dll';
  // disables a login prompt
  IBConnection1.LoginPrompt := False;
  // opens a connection
  IBConnection1.Open;
finally
  IBConnection1.Free;
end;
end;

```

C++ Builder

```

TIBConnection* IBConnection1 = new TIBConnection(NULL);
try {
  // adds connection parameters
  // if Server is empty, a connection is established through the local protocol
  IBConnection1->Server = "server";
  IBConnection1->Database = "database";
  IBConnection1->Username = "username";
  IBConnection1->Password = "password";
  IBConnection1->Port = 3050;
  // indicates the client lib for InterBase (for Firebird, use fbclient.dll)
  IBConnection1->ClientLibrary = "gds32.dll";
  // disables a login prompt
  IBConnection1->LoginPrompt = False;
  // opens a connection
  IBConnection1->Open();
}
finally {
  IBConnection1->Free();
}

```

Opening a Connection

To open a connection at run-time, call the `Open` method:

Delphi

```
IBConnection1.Open;
```

C++Builder

```
IBConnection1->Open;
```

Another way to open a connection at runtime is to set the [Connected](#) property to `True`:

Delphi

```
IBCCConnection1.Connected := True;
```

C++ Builder

```
IBCCConnection1->Connected = True;
```

You can also set up the `Connected` property at design-time in the Object Inspector.

Modifying a Connection

You can modify a connection by changing properties of the `TIBCCConnection` object. Note that while some of the object's properties can be altered without changing the state of a connection, in most cases, a connection is closed when a new value is assigned to the property. For example, if you change the value of the `Server` property, a connection is closed immediately and you need to reopen it manually.

Closing a Connection

To close a connection, call the `Close` method or set the `Connected` property to `False`:

Delphi

```
IBCCConnection1.Close;
```

or:

```
IBCCConnection1.Connected := False;
```

C++ Builder

```
IBCCConnection1->Close;
```

or:

```
IBCCConnection1->Connected = False;
```

Additional Information

IBDAC offers a wide set of features to achieve better performance, balance network load, and enable additional capabilities, for example:

- Local Failover
- Connection Pooling
- Disconnected Mode

- Support for Unicode
- Data Type Mapping

See Also

- [TIBConnection](#)
- [Server](#)
- [Port](#)
- [Database](#)
- [Username](#)
- [Password](#)
- [LoginPrompt](#)
- [ConnectionString](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

3.4 Deleting Data From Tables

This tutorial describes how to delete data from a table using the [TIBCQuery](#) and [TIBCTable](#) components.

1. [Requirements](#)
2. [General Information](#)
3. [Using the DataSet Functionality](#)
4. [Building DML Statements Manually](#)
 - 4.1 [Parameterized DML Statements](#)
 - 4.2 [Non-Parameterized DML Statements](#)
5. [Additional Information](#)

Requirements

This tutorial assumes that you have already connected to the server (see [Connecting to InterBase and Firebird](#)), created the necessary objects on the server (see [Creating Database](#)

[Objects](#)), and inserted data into tables (see [Inserting Data Into Tables](#)). To delete data at runtime, add the [IBC](#) unit to the `uses` clause for Delphi or include the `IBC.hpp` header file for C++ Builder.

General Information

You can delete data from a table using the Data Manipulation Language (DML), which is part of SQL. The user must have the appropriate privileges to execute DML statements on the server. There are two ways to manipulate data in a table: you can build DML statements manually and run them with a component like `TIBCQuery`, or you can use the dataset functionality (the `Delete` method) of the `TIBCQuery` and `TIBCTable` components. Both ways are covered in this tutorial. This tutorial shows you how to delete a record from the [dept](#) table.

Using the DataSet Functionality

The `Delete` method of the `TIBCQuery` and `TIBCTable` components allows you to delete data without having to manually construct a DML statement — it is generated by IBDAC components internally. The code below demonstrates the use of this method:

Delphi

```
var
  IBCQuery1: TIBCQuery;
begin
  IBCQuery1 := TIBCQuery.Create(nil);
  try
    // IBCConnection1 was set up earlier
    IBCQuery1.Connection := IBCConnection1;
    // adds a statement to retrieve data
    IBCQuery1.SQL.Text := 'SELECT * FROM dept';
    // opens the dataset
    IBCQuery1.Open;
    // deletes the active record
    IBCQuery1.Delete;
  finally
    IBCQuery1.Free;
  end;
end;
```

C++Builder

```
TIBCQuery* IBCQuery1 = new TIBCQuery(NULL);
try {
  // IBCConnection1 was set up earlier
  IBCQuery1->Connection = IBCConnection1;
  // adds a statement to retrieve data
  IBCQuery1->SQL->Text = "SELECT * FROM dept";
```

```

// opens the dataset
IBCQuery1->Open();
// deletes the active record
IBCQuery1->Delete();
}
finally {
IBCQuery1->Free();
}

```

Building DML Statements Manually

DML statements can be constructed with or without parameters. The code below demonstrates both ways.

Parameterized DML Statements

Delphi

```

var
  IBCQuery1: TIBCQuery;
begin
  IBCQuery1 := TIBCQuery.Create(nil);
  try
    // IBCConnection1 was set up earlier
    IBCQuery1.Connection := IBCConnection1;
    // adds a statement to delete a record
    IBCQuery1.SQL.Add('DELETE FROM dept WHERE deptno = :deptno;');
    // searches parameters by their names and assigns new values
    IBCQuery1.ParamByName('deptno').AsInteger := 10;
    // executes the statement
    IBCQuery1.Execute;
  finally
    IBCQuery1.Free;
  end;
end;

```

C++Builder

```

TIBCQuery* IBCQuery1 = new TIBCQuery(NULL);
try {
  // IBCConnection1 was set up earlier
  IBCQuery1->Connection = IBCConnection1;
  // adds a statement to delete a record
  IBCQuery1->SQL->Add("DELETE FROM dept WHERE deptno = :deptno;");
  // searches parameters by their names and assigns new values
  IBCQuery1->ParamByName("deptno")->AsInteger = 10;
  // executes the statement
  IBCQuery1->Execute();
}
finally {
  IBCQuery1->Free();
}
}

```

Non-Parameterized DML Statements

Delphi

```
var
  IBCQuery1: TIBCQuery;
begin
  IBCQuery1 := TIBCQuery.Create(nil);
  try
    // IBCConnection1 was set up earlier
    IBCQuery1.Connection := IBCConnection1;
    // adds a statement to delete a record
    IBCQuery1.SQL.Add('DELETE FROM dept WHERE deptno = 10;');
    // executes the statement
    IBCQuery1.Execute;
  finally
    IBCQuery1.Free;
  end;
end;
```

C++Builder

```
TIBCQuery* IBCQuery1 = new TIBCQuery(NULL);
try {
  // IBCConnection1 was set up earlier
  IBCQuery1->Connection = IBCConnection1;
  // adds a statement to delete a record
  IBCQuery1->SQL->Add("DELETE FROM dept WHERE deptno = 10;");
  // executes the statement
  IBCQuery1->Execute();
}
finally {
  IBCQuery1->Free();
}
```

Additional Information

It is also possible to use stored procedures to delete data, in which case all data manipulation logic is defined on the server. See [Using Stored Procedures](#) for more information.

© 1997-2024

Devart. All Rights

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

3.5 Creating Database Objects

This tutorial describes how to create database objects in InterBase and Firebird using the [TIBCSQL](#) and [TIBScript](#) components.

1. [Requirements](#)

2. [General Information](#)
3. [Creating Tables](#)
 - 3.1 [Design-Time](#)
 - 3.2 [Runtime](#)
4. [Creating Stored Procedures](#)
 - 4.1 [Design-Time](#)
 - 4.2 [Runtime](#)
5. [Additional Information](#)

Requirements

This tutorial assumes that you have already connected to the server (see [Connecting to InterBase and Firebird](#)). To create database objects at runtime, add the `IBC` and `IBCScript` units to the `uses` clause for Delphi or include the `IBC.hpp` and `IBCScript.hpp` header files for C++ Builder.

General Information

Database objects are created using Data Definition Language (DDL), which is part of the SQL language. The user must have the appropriate privileges to execute DDL statements on the server. There are two ways to create database objects: build DDL statements manually and execute them with a component like `TIBCSQL`, or use GUI tools for databases. This tutorial uses the data access components to create tables and stored procedures.

Creating Tables

To create tables, the `TIBCSQL` component is used in this tutorial.

Design-Time

- Find the `TIBCSQL` component in the IBDAC category on the Tool Palette.
- Double-click the component. A new object will appear on the form. If this is the first `TIBCSQL` object in this project, it will be named `IBCSQL1`. Note that the [Connection](#) property is automatically set to an existing connection.
- Double-click the `IBCSQL1` object.

- Enter the following statements:

```
CREATE TABLE dept (  
  deptno integer not null primary key,  
  dname varchar(14),  
  loc varchar(13),  
  primary key (deptno)  
);  
CREATE TABLE emp (  
  empno integer not null primary key,  
  ename varchar(10),  
  job varchar(9),  
  mgr integer,  
  hiredate timestamp,  
  sal integer,  
  comm integer,  
  deptno integer references dept (deptno)  
);
```

- Click the `Execute` button to create two tables.

Runtime

The same tables created at runtime:

Delphi

```
var  
  IBSQL1: TIBSQL;  
begin  
  IBSQL1 := TIBSQL.Create(nil);  
  try  
    // IBConnection1 was set up earlier  
    IBSQL1.Connection := IBConnection1;  
    //adds statements to create tables  
    IBSQL1.SQL.Add('CREATE TABLE dept (');  
    IBSQL1.SQL.Add('  deptno integer not null primary key,');  
    IBSQL1.SQL.Add('  dname varchar(14),');  
    IBSQL1.SQL.Add('  loc varchar(13)');  
    IBSQL1.SQL.Add(');');  
    IBSQL1.SQL.Add('CREATE TABLE emp (');  
    IBSQL1.SQL.Add('  empno integer not null primary key,');  
    IBSQL1.SQL.Add('  ename varchar(10),');  
    IBSQL1.SQL.Add('  job varchar(9),');  
    IBSQL1.SQL.Add('  mgr integer,');  
  end try;  
end;
```

```

IBCSQL1.SQL.Add(' hiredate timestamp,');
IBCSQL1.SQL.Add(' sal integer,');
IBCSQL1.SQL.Add(' comm integer,');
IBCSQL1.SQL.Add(' deptno int references dept (deptno)');
IBCSQL1.SQL.Add(');');
// executes the statements
IBCSQL1.Execute;
finally
IBCSQL1.Free;
end;
end;

```

C++Builder

```

TIBCSQL* IBCSQL1 = new TIBCSQL(NULL);
try {
// IBConnection1 was set up earlier
IBCSQL1->Connection = IBConnection1;
//adds statements to create tables
IBCSQL1->SQL->Add("CREATE TABLE dept (");
IBCSQL1->SQL->Add(" deptno integer not null primary key,");
IBCSQL1->SQL->Add(" dname varchar(14),");
IBCSQL1->SQL->Add(" loc varchar(13)");
IBCSQL1->SQL->Add(");");
IBCSQL1->SQL->Add("CREATE TABLE emp (");
IBCSQL1->SQL->Add(" empno integer not null primary key,");
IBCSQL1->SQL->Add(" ename varchar(10),");
IBCSQL1->SQL->Add(" job varchar(9),");
IBCSQL1->SQL->Add(" mgr integer,");
IBCSQL1->SQL->Add(" hiredate timestamp,");
IBCSQL1->SQL->Add(" sal integer,");
IBCSQL1->SQL->Add(" comm integer,");
IBCSQL1->SQL->Add(" deptno integer references dept (deptno)");
IBCSQL1->SQL->Add(");");
// executes the statements
IBCSQL1->Execute();
}
finally {
IBCSQL1->Free();
}

```

Creating Stored Procedures

To create stored procedures, the `TIBCScript` component is used in this tutorial.

Design-Time

- Find the `TIBCScript` component in the `IBDAC` category on the Tool Palette.
- Double-click the component. A new object will appear on the form. If this is the first `TIBCScript` object in this project, it will be named `IBCScript1`. Note that the [Connection](#) property is already set to an existing connection.

- Double-click the `IBCScript1` object.
- Enter the following statement:

```
CREATE PROCEDURE TenMostHighPaidEmployees
RETURNS (salary integer)
AS
BEGIN
  FOR
    SELECT FIRST 10 emp.sal FROM emp ORDER BY emp.sal DESC INTO salary DO
    suspend;
END;
CREATE FUNCTION GetEmpNumberInDept (
  pdeptno integer)
RETURNS integer
AS
BEGIN
  RETURN (SELECT COUNT(*) FROM emp WHERE deptno = :pdeptno);
END;
```

- Click the `Execute` button to create two stored procedures.

Runtime

The same stored procedures created at runtime:

Delphi

```
var
  IBCScript1: TIBCScript;
begin
  IBCScript1 := TIBCScript.Create(nil);
  try
    // IBCConnection1 was set up earlier
    IBCScript1.Connection := IBCConnection1;
    // adds statements to create procedures
    IBCScript1.SQL.Add('CREATE PROCEDURE TenMostHighPaidEmployees');
    IBCScript1.SQL.Add('RETURNS (salary integer)');
    IBCScript1.SQL.Add('AS');
    IBCScript1.SQL.Add('BEGIN');
    IBCScript1.SQL.Add('  FOR');
    IBCScript1.SQL.Add('    SELECT FIRST 10 emp.sal FROM emp ORDER BY emp.sal');
    IBCScript1.SQL.Add('    suspend;');
    IBCScript1.SQL.Add('END');
    IBCScript1.SQL.Add('');
```

```

IBCScript1.SQL.Add('CREATE FUNCTION GetEmpNumberInDept (');
IBCScript1.SQL.Add('  pdeptno integer)');
IBCScript1.SQL.Add('RETURNS integer');
IBCScript1.SQL.Add('AS');
IBCScript1.SQL.Add('BEGIN');
IBCScript1.SQL.Add('  RETURN (SELECT COUNT(*) FROM emp WHERE deptno = :p');
IBCScript1.SQL.Add('END');
// executes the statements
IBCScript1.Execute;
finally
  IBCScript1.Free;
end;
end;

```

C++Builder

```

TIBCScript* IBCScript1 = new TIBCScript(NULL);
try {
  // IBCConnection1 was set up earlier
  IBCScript1->Connection = IBCConnection1;
  // adds statements to create procedures
  IBCScript1->SQL->Add("CREATE PROCEDURE TenMostHighPaidEmployees");
  IBCScript1->SQL->Add("RETURNS (salary integer)");
  IBCScript1->SQL->Add("AS");
  IBCScript1->SQL->Add("BEGIN");
  IBCScript1->SQL->Add("  FOR");
  IBCScript1->SQL->Add("    SELECT FIRST 10 emp.sal FROM emp ORDER BY emp.sa");
  IBCScript1->SQL->Add("    suspend;");
  IBCScript1->SQL->Add("END");
  IBCScript1->SQL->Add("");
  IBCScript1->SQL->Add("CREATE FUNCTION GetEmpNumberInDept (");
  IBCScript1->SQL->Add("  pdeptno integer)");
  IBCScript1->SQL->Add("RETURNS integer");
  IBCScript1->SQL->Add("AS");
  IBCScript1->SQL->Add("BEGIN");
  IBCScript1->SQL->Add("  RETURN (SELECT COUNT(*) FROM emp WHERE deptno =");
  IBCScript1->SQL->Add("END");
  // executes the statements
  IBCScript1->Execute;
}
__finally {
  IBCScript1->Free();
}

```

Additional Information

There are many ways to create database objects on the server. Any tool or component that is capable of running an SQL query can be used to manage database objects. For example, `TIBCSQL` can be used to insert statements one by one, while `TIBCScript` is intended to execute multiple DDL/DML statements as a single SQL script. For more information on DDL statements, refer to the InterBase/Firebird documentation.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

3.6 Inserting Data Into Tables

This tutorial describes how to insert data into tables using the [TIBCQuery](#) and [TIBCTable](#) components.

1. [Requirements](#)
2. [General Information](#)
3. [Design-Time](#)
4. [Runtime](#)
 - 4.1 [Using the DataSet Functionality](#)
 - 4.2 [Building DML Statements Manually](#)
 - 4.2.1 [Parameterized DML Statements](#)
 - 4.2.2 [Non-Parameterized DML Statements](#)
5. [Additional Information](#)

Requirements

This tutorial assumes that you already know how to connect to the server (see [Connecting to InterBase and Firebird](#)) and that the necessary objects have already been created on the server (see [Creating Database Objects](#)). To insert data at runtime, add [IBC](#) unit to the `uses` clause for Delphi or include the `IBC.hpp` header file for C++ Builder.

General Information

You can insert data into a table using the Data Manipulation Language (DML), which is part of SQL. The user must have the appropriate privileges to execute DML statements on the server. There are two ways to manipulate data in a table: you can build DML statements manually and run them with a component like `TIBCQuery`, or you can use the dataset functionality (the `Insert`, `Append`, and `Post` methods) of the `TIBCQuery` and `TIBCTable` components. This tutorial shows you how to insert data into the [dept](#) and [emp](#) tables.

The `dept` table definition:

deptn o	dnam e	loc
10	ACCON UNTIN	NEW YORK

	G	
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

The emp table definition:

emp no	ename	job	mgr	hire date	sal	comm	dept no
7369	SMITH	CLERK	7902	17-12-1980	800	NULL	20
7499	ALLEN	SALESMAN	7698	20-02-1981	1600	300	30
7521	WARD	SALESMAN	7698	22-02-1981	1250	500	30
7566	JONES	MANAGER	7839	02-04-1981	2975	NULL	20
7654	MARTIN	SALESMAN	7698	28-09-1981	1250	1400	30
7698	BLAKE	MANAGER	7839	01-05-1981	2850	NULL	30
7782	CLARK	MANAGER	7839	09-06-1981	2450	NULL	10
7788	SCOTT	ANALYST	7566	13-07-1987	3000	NULL	20
7839	KING	PRESIDENT	NULL	17-11-1981	5000	NULL	10
7844	TURNER	SALESMAN	7698	08-09-1981	1500	0	30

7876	ADAMS	CLERK	7788	13-07-1987	1100	NULL	20
7900	JAMES	CLERK	7698	03-12-1981	950	NULL	30
7902	FORD	ANALYST	7566	03-12-1981	3000	NULL	20
7934	MILLER	CLERK	7782	23-01-1982	1300	NULL	10

Inserting at Design-Time

- Find the `TIBCQuery` component in the `IBDAC` category on the Tool Palette.
- Double-click the component. A new object will appear on the form. If this is the first time that you create a `TIBCQuery` object in this application, the object will be named `IBCQuery1`. Note that the `IBCQuery1.Connection` property is automatically set to an existing connection.
- Double-click the `IBCQuery1` object.
- Enter the following statement:

```
INSERT INTO dept VALUES (10, 'ACCOUNTING', 'NEW YORK');
```

- Click the `Execute` button to add a new record to the `dept` table.

Inserting at Runtime

Using the DataSet Functionality

The `Insert`, `Append`, and `Post` methods of the `TIBCQuery` and `TIBCTable` components allow you to insert data without having to manually construct a DML statement — it is generated by `IBDAC` components internally. `Insert` adds a new empty record in the current cursor position, while `Append` adds a new empty record at the end of the dataset. The code below demonstrates the use of these methods:

Delphi

```
var  
  IBCQuery1: TIBCQuery;  
begin
```

```

IBCQuery1 := TIBCQuery.Create(nil);
try
// IBCConnection1 was set up earlier
  IBCQuery1.Connection := IBCConnection1;
  // adds a statement to retrieve data
  IBCQuery1.SQL.Text := 'SELECT * FROM dept';
  // opens the dataset
  IBCQuery1.Open;
  // adds a new empty record at the end of the dataset
  IBCQuery1.Append;
  // searches fields by their names and assigns new values
  IBCQuery1.FieldByName('deptno').AsInteger := 10;
  IBCQuery1.FieldByName('dname').AsString := 'ACCOUNTING';
  IBCQuery1.FieldByName('loc').AsString := 'NEW YORK';
  // writes the modified record
  IBCQuery1.Post;
finally
  IBCQuery1.Free;
end;
end;

```

C++Builder

```

TIBCQuery* IBCQuery1 = new TIBCQuery(NULL);
try {
// IBCConnection1 was set up earlier
  IBCQuery1->Connection = IBCConnection1;
  // adds a statement to retrieve data
  IBCQuery1->SQL->Text = "SELECT * FROM dept";
  // opens the dataset
  IBCQuery1->Open();
  // adds a new empty record at the end of the dataset
  IBCQuery1->Append();
  // searches fields by their names and assigns new values
  IBCQuery1->FieldByName("deptno")->AsInteger = 10;
  IBCQuery1->FieldByName("dname")->AsString = "ACCOUNTING";
  IBCQuery1->FieldByName("loc")->AsString = "NEW YORK";
  // writes the modified record
  IBCQuery1->Post();
}
__finally {
  IBCQuery1->Free();
}

```

Building DML Statements Manually

DML statements can be constructed with or without parameters. The code below demonstrates both ways.

Parameterized DML Statements

Delphi

```
var
```

```
IBCQuery1: TIBCQuery;
begin
  IBCQuery1 := TIBCQuery.Create(nil);
  try
    // IBCConnection1 was set up earlier
    IBCQuery1.Connection := IBCConnection1;
    // adds a parameterized statement to insert data
    IBCQuery1.SQL.Add('INSERT INTO dept(deptno, dname, loc) VALUES (:deptno,
    // searches parameters by their names and assigns new values
    IBCQuery1.ParamByName('deptno').AsInteger := 10;
    IBCQuery1.ParamByName('dname').AsString := 'ACCOUNTING';
    IBCQuery1.ParamByName('loc').AsString := 'NEW YORK';
    // executes the statement
    IBCQuery1.Execute;
  finally
    IBCQuery1.Free;
  end;
end;
```

C++Builder

```
TIBCQuery* IBCQuery1 = new TIBCQuery(NULL);
try {
  // IBCConnection1 was set up earlier
  IBCQuery1->Connection = IBCConnection1;
  // adds a parameterized statement to insert data
  IBCQuery1->SQL->Add("INSERT INTO dept(deptno, dname, loc) VALUES (:deptno,
  // searches parameters by their names and assigns new values
  IBCQuery1->ParamByName("deptno")->AsInteger = 10;
  IBCQuery1->ParamByName("dname")->AsString = "ACCOUNTING";
  IBCQuery1->ParamByName("loc")->AsString = "NEW YORK";
  // executes the statement
  IBCQuery1->Execute();
}
finally {
  IBCQuery1->Free();
}
```

Non-Parameterized DML Statements

Delphi

```
var
  IBCQuery1: TIBCQuery;
begin
  IBCQuery1 := TIBCQuery.Create(nil);
  try
    // IBCConnection1 was set up earlier
    IBCQuery1.Connection := IBCConnection1;
    // adds a statement to insert a record
    IBCQuery1.SQL.Add('INSERT INTO dept(deptno, dname, loc) VALUES (10, 'ACC
    // executes the statement
    IBCQuery1.Execute;
  finally
    IBCQuery1.Free;
  end;
```

```
end;
```

C++Builder

```
TIBCQuery* IBCQuery1 = new TIBCQuery(NULL);
try {
    // IBCConnection1 was set up earlier
    IBCQuery1->Connection = IBCConnection1;
    //adds the statement to insert a record
    IBCQuery1->SQL->Add("INSERT INTO dept(deptno, dname, loc) VALUES (10,'AC
    // executes the statement
    IBCQuery1->Execute();
}
__finally {
    IBCQuery1->Free();
}
```

Additional Information

There are many ways to insert data into tables. Any tool or component that is capable of running an SQL query can be used to manage data. For example, [TIBCSQL](#) can be used to insert records one by one, while [TIBCScript](#) is designed to execute multiple DDL/DML statements as a single SQL script. [TIBCLoader](#) is the fastest way to insert data into InterBase/Firebird tables.

It is also possible to use stored procedures to insert data, in which case all data manipulation logic is defined on the server. See [Using Stored Procedures](#) for more information.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

3.7 Retrieving Data From Tables

This tutorial describes how to retrieve data from a table using the [TIBCQuery](#) and [TIBCTable](#) components.

1. [Requirements](#)
2. [General Information](#)
3. [TIBCQuery](#)
4. [TIBCTable](#)
5. [Additional Information](#)

Requirements

This tutorial assumes that you have already connected to the server (see [Connecting to InterBase and Firebird](#)), created the necessary database objects (see [Creating Database Objects](#)), and inserted data into tables (see [Inserting Data Into Tables](#)). To retrieve data at runtime, add the `IBC` unit to the `uses` clause for Delphi or include the `IBC.hpp` header file for C++ Builder.

General Information

IBDAC provides the `TIBCQuery` and `TIBCTable` components for retrieving data from a table. This tutorial shows you how to retrieve data from the `dept` table.

TIBCQuery

The following code demonstrates how to retrieve data from the `dept` table using `TIBCQuery`:

Delphi

```
var
  IBCQuery1: TIBCQuery;
begin
  IBCQuery1 := TIBCQuery.Create(nil);
  try
    // IBCConnection1 was set up earlier
    IBCQuery1.Connection := IBCConnection1;
    // adds a statement to retrieve data
    IBCQuery1.SQL.Text := 'SELECT * FROM dept';
    // opens the dataset
    IBCQuery1.Open;
    // shows the number of records in the dataset
    ShowMessage(IntToStr(IBCQuery1.RecordCount));
  finally
    IBCQuery1.Free;
  end;
end;
```

C++Builder

```
TIBCQuery* IBCQuery1 = new TIBCQuery(NULL);
try {
  // IBCConnection1 was set up earlier
  IBCQuery1->Connection = IBCConnection1;
  // adds a statement to retrieve data
  IBCQuery1->SQL->Text = "SELECT * FROM dept";
  // opens the dataset
  IBCQuery1->Open();
  // shows the number of records in the dataset
  ShowMessage(IntToStr(IBCQuery1->RecordCount));
}
```

```
__finally {  
    IBCQuery1->Free();  
}
```

TIBCTable

The following code demonstrates how to retrieve data from the `dept` table using `TIBCTable`:

Delphi

```
var  
    IBCTable1: TIBCTable;  
begin  
    IBCTable1 := TIBCTable.Create(nil);  
    try  
        // IBCTable1 was set up earlier  
        IBCTable1.Connection := IBCTable1.Connection1;  
        // indicates the name of the table  
        IBCTable1.TableName := 'dept';  
        // opens the dataset  
        IBCTable1.Open;  
        // shows the number of records in the dataset  
        ShowMessage(IntToStr(IBCTable1.RecordCount));  
    finally  
        IBCTable1.Free;  
    end;  
end;
```

C++Builder

```
TIBCTable* IBCTable1 = new TIBCTable(NULL);  
try {  
    // IBCTable1 was set up earlier  
    IBCTable1->Connection = IBCTable1->Connection1;  
    // indicates the name of the table  
    IBCTable1->TableName = "dept";  
    // opens the dataset  
    IBCTable1->Open();  
    // shows the number of records in the dataset  
    ShowMessage(IntToStr(IBCTable1->RecordCount));  
}  
__finally {  
    IBCTable1->Free();  
}
```

Additional Information

It is also possible to use stored procedures to delete data, in which case all data manipulation logic is defined on the server. See [Using Stored Procedures](#) for more information.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

3.8 Modifying Data in Tables

This tutorial describes how to modify data in tables using the [TIBCQuery](#) and [TIBCTable](#) components.

1. [Requirements](#)
2. [General Information](#)
3. [Using the DataSet Functionality](#)
4. [Building DML Statements Manually](#)
 - 4.1 [Parameterized DML Statements](#)
 - 4.2 [Non-Parameterized DML Statements](#)
5. [Additional Information](#)

Requirements

This tutorial assumes that you have already connected to the server (see [Connecting to InterBase and Firebird](#)), created the necessary objects on the server (see [Creating Database Objects](#)), and inserted data into tables (see [Inserting Data Into Tables](#)). To modify data at runtime, add the [IBC](#) unit to the `uses` clause for Delphi or include the `IBC.hpp` header file for C++ Builder.

General Information

You can modify data in a table using the Data Manipulation Language (DML), which is part of SQL. DML statements can be executed on the server by a user with respective privileges. There are two ways to manipulate data in a table: you can build DML statements manually and run them with a component like `TIBCQuery`, or you can use the dataset functionality (the `Edit` and `Post` methods) of the `TIBCQuery` and `TIBCTable` components. This tutorial shows you how to modify data in the [dept](#) table:

10	ACCOUNTING	NEW YORK
----	------------	----------

to change it to:

10	RESEARCH	LOS ANGELES
----	----------	-------------

Using the DataSet Functionality

The `Edit` and `Post` methods of the `TIBCQuery` and `TIBCTable` components allows you to modify data without having to manually construct a DML statement — it is generated by IBDAC components internally. The code below demonstrates the use of these methods:

Delphi

```
var
  IBCQuery1: TIBCQuery;
begin
  IBCQuery1 := TIBCQuery.Create(nil);
  try
    // IBCConnection1 was set up earlier
    IBCQuery1.Connection := IBCConnection1;
    // adds a statement to retrieve data
    IBCQuery1.SQL.Text := 'SELECT * FROM dept';
    // opens the dataset
    IBCQuery1.Open;
    // positions the cursor on the deptno=10 record
    IBCQuery1.FindKey([10]);
    // enables editing of data in the dataset
    IBCQuery1.Edit;
    // searches fields by their names and assigns new values
    IBCQuery1.FieldByName('dname').AsString := 'RESEARCH';
    IBCQuery1.FieldByName('loc').AsString := 'LOS ANGELES';
    // writes the modified record
    IBCQuery1.Post;
  finally
    IBCQuery1.Free;
  end;
end;
```

C++Builder

```
TIBCQuery* IBCQuery1 = new TIBCQuery(NULL);
try {
  // IBCConnection1 was set up earlier
  IBCQuery1->Connection = IBCConnection1;
  // adds a statement to retrieve data
  IBCQuery1->SQL->Text = "SELECT * FROM dept";
  // opens the dataset
  IBCQuery1->Open();
  // positions the cursor on the deptno=10 record
  IBCQuery1->FindKey(ARRAYOFCONST((10)));
  // enables editing of data in the dataset
  IBCQuery1->Edit();
  // searches fields by their names and assigns new values
  IBCQuery1->FieldByName("dname")->AsString = "RESEARCH";
  IBCQuery1->FieldByName("loc")->AsString = "LOS ANGELES";
  // writes the modified record
  IBCQuery1->Post();
}
finally {
  q->Free();
}
```

Building DML Statements Manually

DML statements can be constructed with or without parameters. The code below demonstrates both ways.

Parameterized DML Statements

Delphi

```
var
  IBCQuery1: TIBCQuery;
begin
  IBCQuery1 := TIBCQuery.Create(nil);
  try
    // IBCConnection1 was set up earlier
    IBCQuery1.Connection := IBCConnection1;
    // adds a statement to update a record
    IBCQuery1.SQL.Add('UPDATE dept SET dname = :dname, loc = :loc WHERE deptno = :deptno');
    // searches parameters by their names and assigns new values
    IBCQuery1.ParamByName('deptno').AsInteger := 10;
    IBCQuery1.ParamByName('dname').AsString := 'RESEARCH';
    IBCQuery1.ParamByName('loc').AsString := 'LOS ANGELES';
    // executes the statement
    IBCQuery1.Execute;
  finally
    IBCQuery1.Free;
  end;
end;
```

C++Builder

```
TIBCQuery* IBCQuery1 = new TIBCQuery(NULL);
try {
  // IBCConnection1 was set up earlier
  IBCQuery1->Connection = IBCConnection1;
  // adds a statement to update a record
  IBCQuery1->SQL->Add("UPDATE dept SET dname = :dname, loc = :loc WHERE deptno = :deptno");
  // searches parameters by their names and assigns new values
  IBCQuery1->ParamByName("deptno")->AsInteger = 10;
  IBCQuery1->ParamByName("dname")->AsString = "RESEARCH";
  IBCQuery1->ParamByName("loc")->AsString = "LOS ANGELES";
  // executes the statement
  IBCQuery1->Execute();
}
finally {
  IBCQuery1->Free();
}
```

Non-Parameterized DML Statements

Delphi

```
var
```

```

IBCQuery1: TIBCQuery;
begin
  IBCQuery1 := TIBCQuery.Create(nil);
  try
    // IBCConnection1 was set up earlier
    IBCQuery1.Connection := IBCConnection1;
    // adds the statement to update a record
    IBCQuery1.SQL.Add('UPDATE dept SET dname = 'RESEARCH', loc = 'LOS ANGELES');
    // executes the statement
    IBCQuery1.Execute;
  finally
    IBCQuery1.Free;
  end;
end;

```

C++Builder

```

TIBCQuery* IBCQuery1 = new TIBCQuery(NULL);
try {
  // IBCConnection1 was set up earlier
  IBCQuery1->Connection = IBCConnection1;
  // adds a statement to update a record
  IBCQuery1->SQL->Add("UPDATE dept SET dname = 'RESEARCH', loc = 'LOS ANGELES');
  // executes the statement
  IBCQuery1->Execute();
}
finally {
  IBCQuery1->Free();
}

```

Additional Information

It is also possible to use stored procedures to delete data, in which case all data manipulation logic is defined on the server. See [Using Stored Procedures](#) for more information.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

3.9 Using Stored Procedures

This tutorial describes how to work with stored procedures using the [TIBCStoredProc](#) component to insert data into tables.

1. [Requirements](#)
2. [General information](#)
3. [Input Parameters](#)
4. [Output Parameters](#)
5. [Input/Output Parameters](#)

6. [Using Firebird Stored Functions](#)

7. [Returning Result Sets](#)

Requirements

This tutorial assumes that you have already connected to the server (see [Connecting to InterBase and Firebird](#)), created the necessary objects on the server (see [Creating Database Objects](#)), and inserted data into tables (see [Inserting Data Into Tables](#)). To insert data at runtime, add the [IBC](#) unit to the `uses` clause for Delphi or include the `IBC.hpp` header file for C++ Builder.

General Information

A `stored procedure` is a group of one or more SQL statements grouped as a logical unit and stored in the database. Stored procedures are intended to perform a specific task or a set of related tasks. They combine the ease and flexibility of the SQL language with the procedural functionality of a structured programming language. Complicated business rules and programming logic that may require execution of multiple SQL statements should be kept in stored procedures, which can be called by the client applications.

A `stored function` is similar to a stored procedure, but there are some differences: a function must return a value, whereas in a stored procedure it is optional; a function can have only input parameters, whereas a procedure can have input or output parameters; a function can be called from a procedure, whereas a procedure cannot be called from a function.

Input parameters

`Input parameters` are used to pass values from the calling program to the stored procedure. If the procedure changes the input value, the change has effect only within the procedure, and the input variable will preserve its original value when control is returned to the calling program. The following procedure inserts a new row into the table [dept](#):

```
CREATE PROCEDURE InsertDept(  
  p_deptno integer,  
  p_dname varchar(14),  
  p_loc varchar(13)  
) AS  
BEGIN  
  INSERT INTO dept(deptno, dname, loc) VALUES(:p_deptno, :p_dname, :p_loc);  
END;
```

The procedure accepts three input arguments that correspond to the fields of the table, and

can be executed as follows:

```
EXECUTE PROCEDURE InsertDept(10, 'ACCOUNTING', 'NEW YORK');
```

The code below demonstrates the use of the `TIBCStoredProc` component to execute the `InsertDept` stored procedure:

Delphi

```
var
  IBCStoredProc1: TIBCStoredProc;
begin
  IBCStoredProc1 := TIBCStoredProc.Create(nil);
  try
    // IBConnection1 was set up earlier
    IBCStoredProc1.Connection := IBConnection1;
    // indicates the name of the stored procedure to call
    IBCStoredProc1.StoredProcName := 'InsertDept';
    // constructs a statement based on the Params and StoredProcName
    // properties, and assigns it to the SQL property
    IBCStoredProc1.PrepareSQL;
    // searches parameters by their names and assigns new values
    IBCStoredProc1.ParamByName('p_deptno').AsInteger := 10;
    IBCStoredProc1.ParamByName('p_dname').AsString := 'ACCOUNTING';
    IBCStoredProc1.ParamByName('p_loc').AsString := 'NEW YORK';
    // executes the stored procedure
    IBCStoredProc1.Execute;
  finally
    IBCStoredProc1.Free;
  end;
end;
```

C++Builder

```
TIBCStoredProc* IBCStoredProc1 = new TIBCStoredProc(NULL);
try {
  // IBConnection1 was set up earlier
  IBCStoredProc1->Connection = IBConnection1;
  // indicates the name of the stored procedure to call
  IBCStoredProc1->StoredProcName = "InsertDept";
  // constructs a statement based on the Params and StoredProcName
  // properties, and assigns it to the SQL property
  IBCStoredProc1->PrepareSQL();
  // searches parameters by their names and assigns new values
  IBCStoredProc1->ParamByName("p_deptno")->AsInteger = 10;
  IBCStoredProc1->ParamByName("p_dname")->AsString = "ACCOUNTING";
  IBCStoredProc1->ParamByName("p_loc")->AsString = "NEW YORK";
  // executes the stored procedure
  IBCStoredProc1->Execute();
}
finally {
  IBCStoredProc1->Free();
}
```

Output Parameters

Output parameters are used to return values from the procedure to the calling application. The initial value of the parameter in the procedure is NULL, and the value becomes visible to the calling program only when the procedure returns it. The following stored procedure returns the count of records in the [dept](#) table:

```
CREATE PROCEDURE CountDept
RETURNS (cnt integer)
BEGIN
    SELECT count(*) FROM dept into cnt;
END;
```

The code below demonstrates the use of the `TIBCStoredProc` component to execute the `CountDept` stored procedure:

Delphi

```
var
    IBCStoredProc1: TIBCStoredProc;
begin
    IBCStoredProc1 := TIBCStoredProc.Create(nil);
    try
        // IBCCConnection1 was set up earlier
        IBCStoredProc1.Connection := IBCCConnection1;
        // indicates the name of the stored procedure to call
        IBCStoredProc1.StoredProcName := 'CountDept';
        // constructs a statement based on the Params and StoredProcName
        // properties, and assigns it to the SQL property
        IBCStoredProc1.PrepareSQL;
        // executes the stored procedure
        IBCStoredProc1.Execute;
        // shows the value of the output parameter
        ShowMessage(IntToStr(sp.ParamByName('cnt').AsInteger));
    finally
        IBCStoredProc1.Free;
    end;
end;
```

C++Builder

```
TIBCStoredProc* IBCStoredProc1 = new TIBCStoredProc(NULL);
try {
    // IBCCConnection1 was set up earlier
    IBCStoredProc1->Connection = IBCCConnection1;
    // indicates the name of the stored procedure to call
    IBCStoredProc1->StoredProcName = "CountDept";
    // constructs a statement based on the Params and StoredProcName
    // properties, and assigns it to the SQL property
    IBCStoredProc1->PrepareSQL();
    // executes the stored procedure
    IBCStoredProc1->Execute();
    // shows the value of the output parameter
```

```
ShowMessage(IntToStr(sp->ParamByName("cnt")->AsInteger));
}
finally {
    IBCStoredProc1->Free();
}
```

Input/output parameters

A stored procedure that contains input and output parameters can both accept and return values. Programs can pass a value to the stored procedure, which does something under the hood, and passes the resulting value back to the calling program. The input value must be set before executing the stored procedure. The output value is returned after executing the stored procedure.

The following stored procedure returns the salary with a 5% percent bonus:

```
CREATE PROCEDURE GiveBonus(sal float)
RETURNS (bonus float)
AS
BEGIN
    bonus=sal * 1.05;
END;
```

The code below demonstrates the use of the `TIBCStoredProc` component to execute the `GiveBonus` stored procedure:

Delphi

```
var
    IBCStoredProc1: TIBCStoredProc;
begin
    IBCStoredProc1 := TIBCStoredProc.Create(nil);
    try
        // IBCConnection1 was set up earlier
        IBCStoredProc1.Connection := IBCConnection1;
        // indicates the name of the stored procedure to call
        IBCStoredProc1.StoredProcName := 'GiveBonus';
        // constructs a statement based on the Params and StoredProcName
        // properties, and assigns it to the SQL property
        IBCStoredProc1.PrepareSQL;
        // searches a parameter by its name and assigns a new value
        IBCStoredProc1.ParamByName('sal').AsFloat := 500.5;
        // executes the stored procedure
        IBCStoredProc1.Execute;
        // shows the resulting value
        ShowMessage(IBCStoredProc1.ParamByName('sal').AsString);
    finally
        IBCStoredProc1.Free;
    end;
end;
```

C++Builder

```
TIBCStoredProc* IBCStoredProc1 = new TIBCStoredProc(NULL);
try {
    // IBCCConnection1 was set up earlier
    IBCStoredProc1->Connection = IBCCConnection1;
    // indicates the name of the stored procedure to call
    IBCStoredProc1->StoredProcName = "GiveBonus";
    // constructs a statement based on the Params and StoredProcName
    // properties, and assigns it to the SQL property
    IBCStoredProc1->PrepareSQL();
    // searches a parameter by its name and assigns a new value
    IBCStoredProc1->ParamByName("sal")->AsFloat = 500.5;
    // executes the stored procedure
    IBCStoredProc1->Execute();
    // shows the resulting value
    ShowMessage(IBCStoredProc1->ParamByName("sal")->AsString);
}
__finally {
    IBCStoredProc1->Free();
}
```

Using Firebird Stored Functions

The tasks above can also be accomplished using stored functions in Firebird. For example, the following stored function returns the salary with a 5% percent bonus:

```
CREATE FUNCTION GiveBonusFunc(sal float)
RETURNS float
AS
BEGIN
    RETURN sal * 1.05;
END;
```

This function can be executed as follows:

```
SELECT GiveBonusFunc(500.5);
```

The code below demonstrates the use of the `TIBCStoredProc` component to execute the `GiveBonusFunc` stored function:

Delphi

```
var
    IBCStoredProc1: TIBCStoredProc;
begin
    IBCStoredProc1 := TIBCStoredProc.Create(nil);
    try
        // IBCCConnection1 was set up earlier
        IBCStoredProc1.Connection := IBCCConnection1;
        // indicates the name of the stored procedure to call
        IBCStoredProc1.StoredProcName := 'GiveBonusFunc';
        // constructs a statement based on the Params and StoredProcName
        // properties, and assigns it to the SQL property
        IBCStoredProc1.PrepareSQL();
        // searches a parameter by its name and assigns a new value
```

```
IBCStoredProc1.ParamByName('sal').AsFloat := 500.5;
// executes the stored procedure
IBCStoredProc1.Execute;
// shows the resulting value
ShowMessage(IBCStoredProc1.ParamByName('result').AsString);
finally
  IBCStoredProc1.Free;
end;
end;
```

C++Builder

```
TIBCStoredProc* IBCStoredProc1 = new TIBCStoredProc(NULL);
try {
  // IBConnection1 was set up earlier
  IBCStoredProc1->Connection = IBConnection1;
  // indicates the name of the stored procedure to call
  IBCStoredProc1->StoredProcName = "GiveBonusFunc";
  // constructs a statement based on the Params and StoredProcName
  // properties, and assigns it to the SQL property
  IBCStoredProc1->PrepareSQL();
  // searches a parameter by its name and assigns a new value
  IBCStoredProc1->ParamByName("sal")->AsFloat = 500.5;
  // executes the stored procedure
  IBCStoredProc1->Execute();
  // shows the resulting value
  ShowMessage(IBCStoredProc1->ParamByName("result")->AsString);
}
finally {
  IBCStoredProc1->Free();
}
}
```

Note: To retrieve the result returned by the stored function using `TIBCStoredProc`, use the automatically created 'result' parameter.

Returning Result Sets

Besides scalar variables, a stored procedure can return a result set generated by the SELECT statement. See [Using Stored Procedures with Result Sets](#) for more information.

© 1997-2024

Devart. All Rights

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

3.10 Using Stored Procedures with Result Sets

This tutorial describes how to retrieve and modify result sets obtained from stored procedures using the [TIBCStoredProc](#) component

Requirements

This tutorial assumes that you have already connected to the server (see [Connecting to InterBase and Firebird](#)), created the necessary objects on the server (see [Creating Database Objects](#)), and inserted data into tables (see [Inserting Data Into Tables](#)). To insert data at runtime, add the `IBC` unit to the `uses` clause for Delphi or include the `IBC.hpp` header file for C++ Builder.

General Information

Besides scalar variables, a stored procedure can return a result set generated by the `SELECT` statement. You can insert or modify data in a result set using the dataset functionality of the `TIBCStoredProc` component.

This tutorial shows you how to retrieve and modify data in the `dept` table using the `TIBCStoredProc` component. The following stored procedure will be used to retrieve data:

```
CREATE PROCEDURE SelectDept()  
BEGIN  
    SELECT * FROM dept;  
END;
```

Using the DataSet Functionality

The `Insert`, `Append`, `Edit`, and `Post` methods of the `TIBCStoredProc` component can be used to insert and modify data without having to manually construct a DML statement — it is generated by `IBDAC` components internally. The code below demonstrates the use of these methods:

Delphi

```
var  
    IBCStoredProc1: TIBCStoredProc;  
begin  
    IBCStoredProc := TIBCStoredProc.Create(nil);  
    try  
        // IBConnection1 was set up earlier  
        IBCStoredProc.Connection := IBConnection1;  
        // indicates the name of the stored procedure to call  
        IBCStoredProc.StoredProcName := 'SelectDept';  
        // constructs a statement based on the Params and StoredProcName  
        // properties, and assigns it to the SQL property  
        IBCStoredProc.PrepareSQL;  
        // opens the dataset  
        IBCStoredProc.Open;  
        // adds a new empty record at the end of the dataset  
        IBCStoredProc.Append;  
        // searches fields by their names and assigns new values
```

```

IBCStoredProc.FieldName('deptno').AsInteger := 50;
IBCStoredProc.FieldName('dname').AsString := 'SALES';
IBCStoredProc.FieldName('loc').AsString := 'NEW YORK';
// writes the modified record
IBCStoredProc.Post;
// adds a new empty record in the current cursor position
IBCStoredProc.Insert;
IBCStoredProc.FieldName('deptno').AsInteger := 60;
IBCStoredProc.FieldName('dname').AsString := 'ACCOUNTING';
IBCStoredProc.FieldName('loc').AsString := 'LOS ANGELES';
IBCStoredProc.Post;
// positions the cursor on the deptno=10 record
IBCStoredProc.FindKey([10]);
// enables editing of data in the dataset
IBCStoredProc.Edit;
IBCStoredProc.FieldName('dname').AsString := 'RESEARCH';
IBCStoredProc.FieldName('loc').AsString := 'LOS ANGELES';
IBCStoredProc.Post;
finally
  IBCStoredProc.Free;
end;
end;

```

C++Builder

```

TIBCStoredProc* IBCStoredProc = new TIBCStoredProc(NULL);
try {
  // IBCConnection1 was set up earlier
  IBCStoredProc->Connection = IBCConnection1;
  // indicates the name of the stored procedure to call
  IBCStoredProc->StoredProcName = "selectDept";
  // constructs a statement based on the Params and StoredProcName
  // properties, and assigns it to the SQL property
  IBCStoredProc->PrepareSQL();
  // opens the dataset
  IBCStoredProc->Open();
  // adds a new empty record at the end of the dataset
  IBCStoredProc->Append();
  // searches fields by their names and assigns new values
  IBCStoredProc->FieldName("deptno")->AsInteger = 50;
  IBCStoredProc->FieldName("dname")->AsString = "SALES";
  IBCStoredProc->FieldName("loc")->AsString = "NEW YORK";
  // writes the modified record
  IBCStoredProc->Post();
  // adds a new empty record in the current cursor position
  IBCStoredProc->Insert();
  IBCStoredProc->FieldName("deptno")->AsInteger = 60;
  IBCStoredProc->FieldName("dname")->AsString = "ACCOUNTING";
  IBCStoredProc->FieldName("loc")->AsString = "LOS ANGELES";
  IBCStoredProc->Post();
  // positions the cursor on the deptno=10 record
  IBCStoredProc->FindKey(ARRAYOFCONST((10)));
  // enables editing of data in the dataset
  IBCStoredProc->Edit();
  IBCStoredProc->FieldName("dname")->AsString = "RESEARCH";
  IBCStoredProc->FieldName("loc")->AsString = "LOS ANGELES";
  IBCStoredProc->Post();
}

```

```
}  
_finally {  
    IBCStoredProc->Free();  
}
```

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

3.11 Demo Projects

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

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

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

Where are the IBDAC demo projects located?

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

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

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

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

Demos

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

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

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

Instructions for using the IBDAC demo projects

To explore an IBDAC demo project,

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

The executed version of the demo will contain a sample application written with IBDAC or a navigable list of samples and sample descriptions. To use each sample properly, you will need to connect to a working InterBase/Firebird server.

The included sample applications are fully functional. To use the demos, you have to first set up a connection to InterBase. You can do so by clicking on the "Connect" button.

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

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

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

Demo project descriptions

IbDacDemo

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

Working with components

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

General demos

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

InterBase-specific demos

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

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

Working with Components

Name	Description
Alerter	Uses the TIBCAlerter component to send messages between connections using InterBase events.
ConnectDialog	Demonstrates how to customize the IBDAC connect dialog . Changes the standard IBDAC connect dialog to two custom connect dialogs. The first customized sample dialog is inherited from the TForm class, and the second one is inherited from the default IBDAC connect dialog class.
CRDBGrid	Demonstrates how to work with the TCRDBGrid component. Shows off the main TCRDBGrid features, like filtering, searching, stretching, using compound headers, and more.
Query	Demonstrates working with TIBCQuery , which is one of the most useful IBDAC components. Includes many TIBCQuery usage scenarios. Demonstrates how to execute queries, how to edit data

	and export it to XML files. Note: This is a very good introductory demo. We recommend starting here when first becoming familiar with IBDAC.
Sql	Uses TIBCSQL to execute SQL statements. Demonstrates how to prepare the statement and how to work with parameters in SQL.
StoredProc	Uses TIBCStoredProc to access an editable recordset retrieved by an InterBase stored procedure in the client application.
Table	Demonstrates how to use TIBCTable 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.
UpdateSQL	Demonstrates using the TIBCUpdateSQL component to customize update commands. Lets you optionally use TIBCSQL and TIBCQuery 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
CachedUpdates	Demonstrates how to perform the most important tasks of working with data in CachedUpdates mode, including highlighting uncommitted changes, managing transactions, and committing changes in a batch.
FilterAndIndex	Demonstrates IBDAC's local storage functionality. This sample shows how to perform local filtering, sorting and locating by multiple fields, including by calculated and lookup fields.
MasterDetail	Uses IBDAC functionality to work with master/detail relationships . This sample shows how to use local master/detail functionality. Demonstrates different kinds of master/detail linking, including linking by SQL, by simple fields, and by calculated fields.
Threads	Demonstrates how IBDAC can be used in multithreaded applications. This sample allows you to set up several threads and test IBDAC's performance with multithreading.

InterBase-specific Demos

Name	Description
Arrays	Demonstrates working with InterBase arrays . This sample lets you

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

Supplementary Demo Projects

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

Location	Name	Description
ThirdParty	FastReport	Demonstrates how IBDAC can be used with FastReport components. This project consists of two parts. The first part is several packages that integrate IBDAC components into the FastReport editor. The second part is a demo application that lets you design and preview reports with IBDAC technology in the FastReport editor.
	InfoPower	Uses InfoPower components to display recordsets retrieved with IBDAC. This demo project displays an InfoPower grid component and fills it with the result of an IBDAC query. Shows how to link IBDAC data sources to

		InfoPower components.
	IntraWeb	A collection of sample projects that show how to use IBDAC components as data sources for IntraWeb applications. Contains IntraWeb samples for setting up a connection, querying a database and modifying data, and working with CachedUpdates and MasterDetail relationships.
	ReportBuilder	Uses IBDAC data sources to create a ReportBuilder report that takes data from InterBase database. Shows how to set up a ReportBuilder document in design-time and how to integrate IBDAC components into the Report Builder editor to perform document design in run-time.
Miscellaneous	CBuilder	A general demo project about creating IBDAC-based applications with C++Builder. Lets you execute SQL scripts and work with result sets in a grid. This is one of the two IBDAC demos for C++Builder.
	FailOver	Demonstrates the recommended approach to working with unstable networks . 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.
	Midas	Demonstrates using MIDAS technology with IBDAC. This project consists of two parts: a MIDAS server that processes requests to the database and a

		thin MIDAS client that displays an interactive grid. This demo shows how to build thin clients that display interactive components and delegate all database interaction to a server application for processing.
	VirtualTableCB	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. This is one of the two demo projects for C++Builder.
<i>IbDacDemo</i>	<i>IbDacDemo</i>	<i>[Win32 version of the main IBDAC demo project - see above]</i>

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

3.12 Deployment

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

Deploying Windows applications built without run-time packages

You do not need to deploy any files with IBDAC-based applications built without run-time packages, provided you are using a registered version of IBDAC.

You can check if your application does not require run-time packages by making sure the "Build with runtime packages" check box is not selected in the Project Options dialog box.

Trial Limitation Warning

If you are evaluating deploying Windows applications with IBDAC Trial Edition, you will need to deploy the following DAC BPL files:

dacXX.bpl	always
ibdacXX.bpl	always

and their dependencies (required IDE BPL files) with your application, even if it is built without run-time packages:

rtlXX.bpl	always
dbrtlXX.bpl	always
vcldbXXX.bpl	always

Deploying Windows 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 Windows application:

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

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4 Using IBDAC

This section describes basics of using InterBase Data Access Components

- [Updating Data with IBDAC Dataset Components](#)
- [Master/Detail Relationships](#)
- [Automatic Key Field Value Generation](#)

- [Data Type Mapping](#)
- [OTW Encryption](#)
- [Data Encryption](#)
- [Working in an Unstable Network](#)
- [Disconnected Mode](#)
- [Increasing Performance](#)
- [Macros](#)
- [DataSet Manager](#)
- [Connection Pooling](#)
- [BLOB Data Types](#)
- [Unicode Character Data](#)
- [ARRAY Data Type](#)
- [TIBCQuery Component](#)
- [DBMonitor](#)
- [Writing GUI Applications with IBDAC](#)
- [Compatibility with Previous Versions](#)
- [64-bit Development with Embarcadero RAD Studio XE2](#)
- [Database Specific Aspects of 64-bit Development](#)
- [Demo Projects](#)
- [Deployment](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.1 Updating Data with IBDAC Dataset Components

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

The following components are used to execute SQL statements: TIBCQuery,

TIBCStoredProc, TIBCTable.

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

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

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

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

Set the [Transaction](#) property of your DataSet component to the transaction component with ReadCommitted/ReadOnly [IsolationLevel](#) property, and [UpdateTransaction](#) property to the transaction component with ReadCommitted [IsolationLevel](#) property for the optimal transaction using performance. Borland recommends to start the read-only transaction and commit it with [CommitRetaining](#) on InterBase 7.1. Using transactions in such a way minimizes server load.

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

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

Example:

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

After executing or preparing such statement the following out parameters appear: RET_F1 and RET_F2. They will contain values of F1 and F2 fields.

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

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

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.2 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 IBDAC implements this feature.

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

"Department" table has field Dept_No. Dept_No is a primary key.

"Employee" table has a primary key EmpNo and foreign key Dept_No that binds "Employee" to "Department".

It is necessary to display and edit these tables.

IBDAC provides two ways to bind tables. First code example shows how to bind two TCustomIBCDataset components (TIBCQuery or TIBCTable) into MD relationship via parameters.

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

Pay attention to one thing: parameter name in detail dataset SQL must be equal to the field name or the alias in the master dataset that is used as foreign key for detail table. After opening detail dataset always holds records with Dept_No field value equal to the one in the current master dataset record.

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

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

Now suppose that detail table "Department" foreign key field is named DepLink but not Dept_No. In such case detail dataset described in above code example will not autofill DepLink field with current "Department".Dept_No value on insert. This issue is solved in

second code example.

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

In this code example MD relationship is set up using [MasterFields](#) and [DetailFields](#) properties. Also note that there are no WHERE clause in detail dataset SQL.

To defer refreshing of detail dataset while master dataset navigation you can use [DetailDelay](#) option.

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

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.3 Automatic Key Field Value Generation

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

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

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

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

Example:

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

See also

- [KeyGenerator](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.4 Data Type Mapping

Overview

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

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

Data Type Mapping Rules

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

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

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

And Data Type Mapping should be used so that:

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

The above in the form of a table:

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

To specify that numeric fields with Precision <= 4 and Scale = 0 must be mapped to

ftSmallint, such a rule should be set:

```
var
  DBType: word;
  MinPrecision: Integer;
  MaxPrecision: Integer;
  MinScale: Integer;
  MaxScale: Integer;
  FieldType: TFieldType;
begin
  DBType      := ibcNumeric;
  MinPrecision := 10;
  MaxPrecision := r1Any;
  MinScale    := 1;
  MaxScale    := 4;
  FieldType   := ftBCD;
  IBCConnection.DataTypeMap.AddDBTypeRule(DBType, MinPrecision, MaxPrecision,
  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:

```
// rule for numeric(5,2)
IBCConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 0, 9, 1, r1Any, ftF
// rule for numeric(10,4)
IBCConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 10, r1Any, 1, 4, ftB
// rule for numeric(15,6)
IBCConnection.DataTypeMap.AddDBTypeRule(ibcNumeric, 10, r1Any, 5, r1Any, ftF
```

Rules order

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

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

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

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

InterBase or Firebird data	Default Delphi field type	Destination field type
----------------------------	---------------------------	------------------------

type		
NUMBER(5,2)	ftFloat	ftFloat
NUMBER(10,4)	ftFloat	ftBCD
NUMBER(15,6)	ftFloat	ftFMTBCD

If rules are set in the following way:

```
IBCConnection.DataTypeMap.Clear;
IBCConnection.DataTypeMap.AddDBTypeRule(IBC_NUMERIC, 0, 9, r1Any, r1Any,
IBCConnection.DataTypeMap.AddDBTypeRule(IBC_NUMERIC, 0, r1Any, 0, 4,
IBCConnection.DataTypeMap.AddDBTypeRule(IBC_NUMERIC, 0, r1Any, 0, r1Any,
```

it will lead to the following result:

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

But if rules are set in the following way:

```
IBCConnection.DataTypeMap.Clear;
IBCConnection.DataTypeMap.AddDBTypeRule(IBC_NUMERIC, 0, r1Any, 0, r1Any,
IBCConnection.DataTypeMap.AddDBTypeRule(IBC_NUMERIC, 0, r1Any, 0, 4,
IBCConnection.DataTypeMap.AddDBTypeRule(IBC_NUMERIC, 0, 9, r1Any, r1Any,
```

it will lead to the following result:

InterBase data type	Delphi field type
NUMBER(5,2)	ftFMTBCD
NUMBER(10,4)	ftFMTBCD
NUMBER(15,6)	ftFMTBCD

This happens because the rule

```
IBCConnection.DataTypeMap.AddDBTypeRule(IBC_NUMERIC, 0, r1Any, 0, r1Any,
```

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

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

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

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

Defining rules for Connection and Dataset

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

For example, such table is created in SQL Server:

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

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

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

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

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

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

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

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

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

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

Rules for a particular field

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

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

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

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

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

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

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

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

Rules for a particular character set

In InterBase/Firebird, character fields (CHAR, VARCHAR), as well as BLOB fields, may have a charset different from the one of the database. For example, a table may be created in the following way:

```
CREATE TABLE TEST
(
  ID INTEGER NOT NULL PRIMARY KEY,
  FIELD1 VARCHAR(10) CHARACTER SET UTF8,
  FIELD2 VARCHAR(10) CHARACTER SET WIN1251
)
```

To set different mapping-by-charset rules for fields with the same type (e.g., map UTF-8 VARCHAR fields as ftString, and fields encoded in WIN1251 - as ftWideString), you can use the following IBDAC methods:

```
IBCConnection.DataTypeMap.Clear;
IBCConnection.DataTypeMap.AddCharsetRule(IBCVarChar, 'UTF8', ftString);
IBCConnection.DataTypeMap.AddCharsetRule(IBCVarChar, 'WIN1251', ftWideString);
IBCQuery.DataTypeMap.Clear;
IBCQuery.DataTypeMap.AddCharsetRule(IBCVarChar, 'UTF8', ftString);
IBCQuery.DataTypeMap.AddCharsetRule(IBCVarChar, 'WIN1251', ftWideString);
```

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:

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

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

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

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.5 OTW Network Encryption

Encrypting Network Using Over-the-Wire (OTW)

IBDAC supports the Over-the-Wire (OTW) encryption feature of InterBase to encrypt data during the transmission process. InterBase OTW encryption uses SSL v3 and TLS v1 security protocols and supports AES and DES encryptions. Before setting up OTW encryption on the server and client side, you must obtain the necessary security certificates from a certificate authority (CA). Both the client and server must have the X.509 files in the PEM format installed to use OTW encryption. After configuring the OTW parameters on the server, set up the client side in your IBDAC-based application. The OTW encryption parameters can be set up at runtime as follows:

Delphi

```
var
  IBConnection1: TIBConnection;
begin
  IBConnection1 := TIBConnection.Create(nil);
```

```
try
  IBConnection1.Server := '127.0.0.1';
  IBConnection1.Database := 'database';
  IBConnection1.Username := 'username';
  IBConnection1.Password := 'password';
  IBConnection1.Port := 3050;
  IBConnection1.ClientLibrary := 'gds32.dll';
  IBConnection1.LoginPrompt := False;
  // OTW encryption properties
  IBConnection1.SSLOptions.ClientCertFile := 'clientcert.pem';
  IBConnection1.SSLOptions.ClientPassPhrase := 'passphrase';
  IBConnection1.SSLOptions.ServerPublicFile := 'cacert.pem';
  IBConnection1.SSLOptions.Enabled := True;
  IBConnection1.Open;
finally
  IBConnection1.Free;
end;
end;
```

C++ Builder

```
TIBConnection* IBConnection1 = new TIBConnection(NULL);
try {
  IBConnection1->Server = "127.0.0.1";
  IBConnection1->Database = "database";
  IBConnection1->Username = "username";
  IBConnection1->Password = "password";
  IBConnection1->Port = 3050;
  IBConnection1->ClientLibrary = "gds32.dll";
  IBConnection1->LoginPrompt = False;
  // OTW encryption properties
  IBConnection1->SSLOptions->ClientCertFile = "clientcert.pem";
  IBConnection1->SSLOptions->ClientPassPhrase = "passphrase";
  IBConnection1->SSLOptions->ServerPublicFile = "cacert.pem";
  IBConnection1->SSLOptions->Enabled = True;
  IBConnection1->Open();
}
__finally {
  IBConnection1->Free();
}
```

See Also

- [Encrypting Network Communication](#)
- [TIBCSSLConnectionOptions](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.6 Data Encryption

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

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

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

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

To avoid these problems, it is recommended to store, along with the data, the appropriate GUID, which is necessary for specifying that the value in the record is encrypted and it must be decrypted when reading data. This allows you to avoid confusion and keep in the same column both the encrypted and decrypted data, which is particularly important when using an existing table. Also, when doing in this way, a random initializing vector is generated before the data encryption, which is used for encryption. This allows you to receive different results for the same initial data, which significantly increases security.

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

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

auxiliary information.

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

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

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

Example.

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

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

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

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

```
IBCQuery.DataTypeMap.AddFieldNameRule ('SAL', ftFloat);  
IBCQuery.Open;
```

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.7 Working in an Unstable Network

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

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

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

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

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

These settings allow to work with Blobs and Arrays without an active connection.

These settings minimize the number of requests to the server. Using [TCustomDAConnection.Options.DisconnectedMode](#) allows DataSet to work without an active connection. It minimizes server resource usage and reduces connection break probability. I. e. in this mode connection automatically closes if it is not required any more. But every explicit operation must be finished explicitly. That means each explicit connect must be followed by explicit disconnect. Read [Working with Disconnected Mode](#) topic for more information.

Setting the [FetchAll](#) property to True allows to fetch all data after cursor opening and to close

connection. If you are using master/detail relationship, we recommend to set the [LocalMasterDetail](#) option to True.

It is not recommended to prepare queries explicitly. Use the [CachedUpdates](#) mode for DataSet data editing. Use the [TCustomDADataSet.Options.UpdateBatchSize](#) property to reduce the number of requests to the server.

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

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

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

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

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

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

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.8 Disconnected Mode

In disconnected mode a connection opens only when it is required. After performing all server calls connection closes automatically until next server call is required. Datasets remain

opened when connection closes. Disconnected Mode may be useful for saving server resources and operating in an unstable or expensive network. Drawback of using disconnected mode is that each connection establishing requires some time for authorization. If connection is often closed and opened it can slow down application work. We recommend to use pooling to solve this problem. For additional information see [TCustomDACConnection.Pooling](#).

To enable disconnected mode set [TCustomDACConnection.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 [Disonnect](#) method is called or the [Connected](#) property is set to False explicitly.

The following settings are recommended to use for working in disconnected mode:

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

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

These settings minimize the number of requests to the server.

Disconnected mode features

If you perform a query with the [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

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.9 Batch Operations

Data amount processed by modern databases grows steadily. In this regard, there is an acute problem – database performance. Insert, Update and Delete operations have to be performed as fast as possible. Therefore Devart provides several solutions to speed up processing of huge amounts of data. So, for example, insertion of a large portion of data to a DB is supported in the [TIBCLoader](#). Unfortunately, [TIBCLoader](#) allows to insert data only – it can't be used for updating and deleting data.

The new version of Devart Delphi Data Access Components introduces the new mechanism for large data processing — Batch Operations. The point is that just one parametrized Modify SQL query is executed. The plurality of changes is due to the fact that parameters of such a query will be not single values, but a full array of values. Such approach increases the speed of data operations dramatically. Moreover, in contrast to using [TIBCLoader](#), Batch operations can be used not only for insertion, but for modification and deletion as well.

Let's have a better look at capabilities of Batch operations with an example of the BATCH_TEST table containing attributes of the most popular data types.

Batch_Test table generating script

```
CREATE TABLE BATCH_TEST  
(
```

```

ID    INTEGER NOT NULL PRIMARY KEY,
F_INTEGER INTEGER,
F_FLOAT  FLOAT,
F_STRING VARCHAR(250),
F_DATE   DATE
)

```

Batch operations execution

To insert records into the BATCH_TEST table, we use the following SQL query:

```
INSERT INTO BATCH_TEST VALUES (:ID, :F_INTEGER, :F_FLOAT, :F_STRING, :F_DATE)
```

When a simple insertion operation is used, the query parameter values look as follows:

Parameters				
:ID	:F_INTEGER	:F_FLOAT	:F_STRING	:F_DATE
1	100	2.5	'String Value 1'	01.09.2015

After the query execution, one record will be inserted into the BATCH_TEST table.

When using Batch operations, the query and its parameters remain unchanged. However, parameter values will be enclosed in an array:

Parameters				
:ID	:F_INTEGER	:F_FLOAT	:F_STRING	:F_DATE
1	100	2.5	'String Value 1'	01.09.2015
2	200	3.15	'String Value 2'	01.01.2000
3	300	5.08	'String Value 3'	09.09.2010
4	400	7.5343	'String Value 4'	10.10.2015
5	500	0.4555	'String Value 5'	01.09.2015

Now, 5 records are inserted into the table at a time on query execution.

How to implement a Batch operation in the code?

Batch INSERT operation sample

Let's try to insert 1000 rows to the BATCH_TEST table using a Batch Insert operation:

```

var
  i: Integer;
begin
  // describe the SQL query
  IBCQuery1.SQL.Text := 'INSERT INTO BATCH_TEST VALUES (:ID, :F_INTEGER, :F_FLOAT, :F_STRING, :F_DATE)';
  // define the parameter types passed to the query :
  IBCQuery1.Params[0].DataType := ftInteger;

```

```

IBCQuery1.Params[1].DataType := ftInteger;
IBCQuery1.Params[2].DataType := ftFloat;
IBCQuery1.Params[3].DataType := ftString;
IBCQuery1.Params[4].DataType := ftDateTime;
// specify the array dimension:
IBCQuery1.Params.ValueCount := 1000;
// populate the array with parameter values:
for i := 0 to IBCQuery1.Params.ValueCount - 1 do begin
  IBCQuery1.Params[0][i].AsInteger := i + 1;
  IBCQuery1.Params[1][i].AsInteger := i + 2000 + 1;
  IBCQuery1.Params[2][i].AsFloat := (i + 1) / 12;
  IBCQuery1.Params[3][i].AsString := 'Values ' + IntToStr(i + 1);
  IBCQuery1.Params[4][i].AsDateTime := Now;
end;
// insert 1000 rows into the BATCH_TEST table
IBCQuery1.Execute(1000);
end;

```

This command will insert 1000 rows to the table with one SQL query using the prepared array of parameter values. The number of inserted rows is defined in the `Iters` parameter of the `Execute(Iters: integer; Offset: integer = 0)` method. In addition, you can pass another parameter – `Offset` (0 by default) – to the method. The `Offset` parameter points the array element, which the Batch operation starts from.

We can insert 1000 records into the BATCH_TEST table in 2 ways.

All 1000 rows at a time:

```
Query1.Execute(1000);
```

2×500 rows:

```

// insert first 500 rows
IBCQuery1.Execute(500, 0);
// insert next 500 rows
IBCQuery1.Execute(500, 500);

```

500 rows, then 300, and finally 200:

```

// insert 500 rows
Query1.Execute(500, 0);
// insert next 300 rows starting from 500
IBCQuery1.Execute(300, 500);
// insert next 200 rows starting from 800
Query1.Execute(200, 800);

```

Batch UPDATE operation sample

With Batch operations we can modify all 1000 rows of our BATCH_TEST table just this simple:

```

var
  i: Integer;

```

```

begin
  // describe the SQL query
  IBCQuery1.SQL.Text := 'UPDATE BATCH_TEST SET F_INTEGER=:F_INTEGER, F_FLOAT=:F_FLOAT';
  // define parameter types passed to the query:
  IBCQuery1.Params[0].DataType := ftInteger;
  IBCQuery1.Params[1].DataType := ftFloat;
  IBCQuery1.Params[2].DataType := ftString;
  IBCQuery1.Params[3].DataType := ftDateTime;
  IBCQuery1.Params[4].DataType := ftInteger;
  // specify the array dimension:
  IBCQuery1.Params.ValueCount := 1000;
  // populate the array with parameter values:
  for i := 0 to 1000 - 1 do begin
    IBCQuery1.Params[0][i].AsInteger := i - 2000 + 1;
    IBCQuery1.Params[1][i].AsFloat := (i + 1) / 100;
    IBCQuery1.Params[2][i].AsString := 'New Values ' + IntToStr(i + 1);
    IBCQuery1.Params[3][i].AsDateTime := Now;
    IBCQuery1.Params[4][i].AsInteger := i + 1;
  end;
  // update 1000 rows in the BATCH_TEST table
  IBCQuery1.Execute(1000);
end;

```

Batch DELETE operation sample

Deleting 1000 rows from the BATCH_TEST table looks like the following operation:

```

var
  i: Integer;
begin
  // describe the SQL query
  IBCQuery1.SQL.Text := 'DELETE FROM BATCH_TEST WHERE ID=:ID';
  // define parameter types passed to the query:
  IBCQuery1.Params[0].DataType := ftInteger;
  // specify the array dimension
  IBCQuery1.Params.ValueCount := 1000;
  // populate the arrays with parameter values
  for i := 0 to 1000 - 1 do
    IBCQuery1.Params[0][i].AsInteger := i + 1;
  // delete 1000 rows from the BATCH_TEST table
  IBCQuery1.Execute(1000);
end;

```

Performance comparison

The example with BATCH_TEST table allows to analyze execution speed of normal operations with a database and Batch operations:

Operation Type	25 000 records	
	Standard Operation (sec.)	Batch Operation (sec.)
Insert	55.4	3.03

Update	81.9	3.58	
Delete	61.3	0.91	
The less, the better.			

It should be noted, that the retrieved results may differ when modifying the same table on different database servers. This is due to the fact that operations execution speed may differ depending on the settings of a particular server, its current workload, throughput, network connection, etc.

Thing you shouldn't do when accessing parameters in Batch operations!

When populating the array and inserting records, we accessed query parameters by index. It would be more obvious to access parameters by name:

```
for i := 0 to 9999 do begin
  IBCQuery1.Params.ParamByName('ID')[i].AsInteger := i + 1;
  IBCQuery1.Params.ParamByName('F_INTEGER')[i].AsInteger := i + 2000 + 1;
  IBCQuery1.Params.ParamByName('F_FLOAT')[i].AsFloat := (i + 1) / 12;
  IBCQuery1.Params.ParamByName('F_STRING')[i].AsString := 'Values ' + IntToStr(i);
  IBCQuery1.Params.ParamByName('F_DATE')[i].AsDateTime := Now;
end;
```

However, the parameter array would be populated slower, since you would have to define the ordinal number of each parameter by its name in each loop iteration. If a loop is executed 10000 times – **performance loss can become quite significant.**

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.10 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 (`TCustomDACConnection.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 [TCustomDAUpdateSQL](#). The performance gain achieved this way can be anywhere from several percent to several times, depending on the situation.

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

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

Fetch

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

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

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

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

Navigate

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

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

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

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

Update

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

Specifying update SQL statements in a dataset improves performance because of omitting SQL statements generation and automatic preparation of internal updating datasets that are created for every kind of update SQL statements.

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.11 Macros

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

First step is to assign macros with their names and values to a dataset object.

Then modify SQL statement to include macro names into desired insertion points. Prefix

each name with & ("at") sign to let IBDAC discriminate them at parse time. Resolved SQL statement will hold macro values instead of their names but at the right places of their occurrences. For example, having the following statement with the TableName macro name:

```
SELECT * FROM &TableName
```

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

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

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

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

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

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

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

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

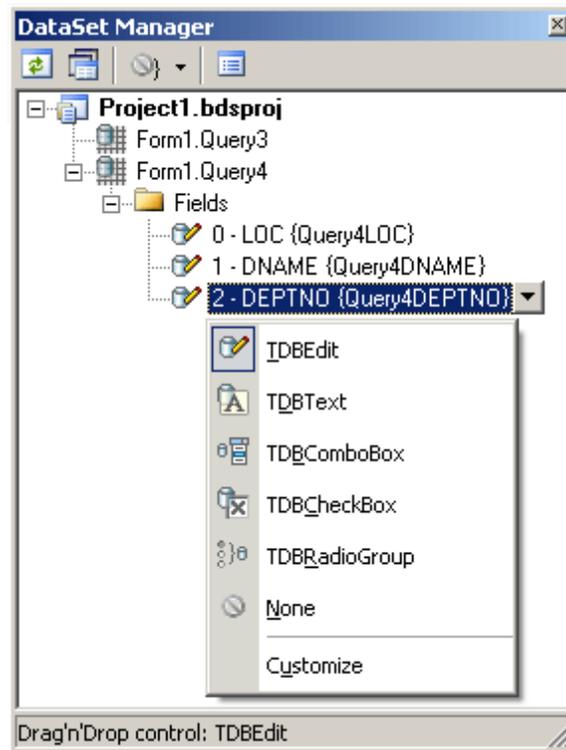
[DAC Forum](#)

[Provide Feedback](#)

4.12 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.

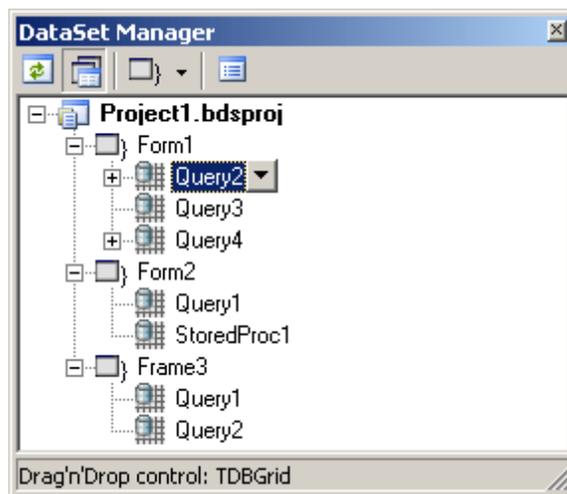
Opening the DataSet Manager window

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 Delphi 6 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.

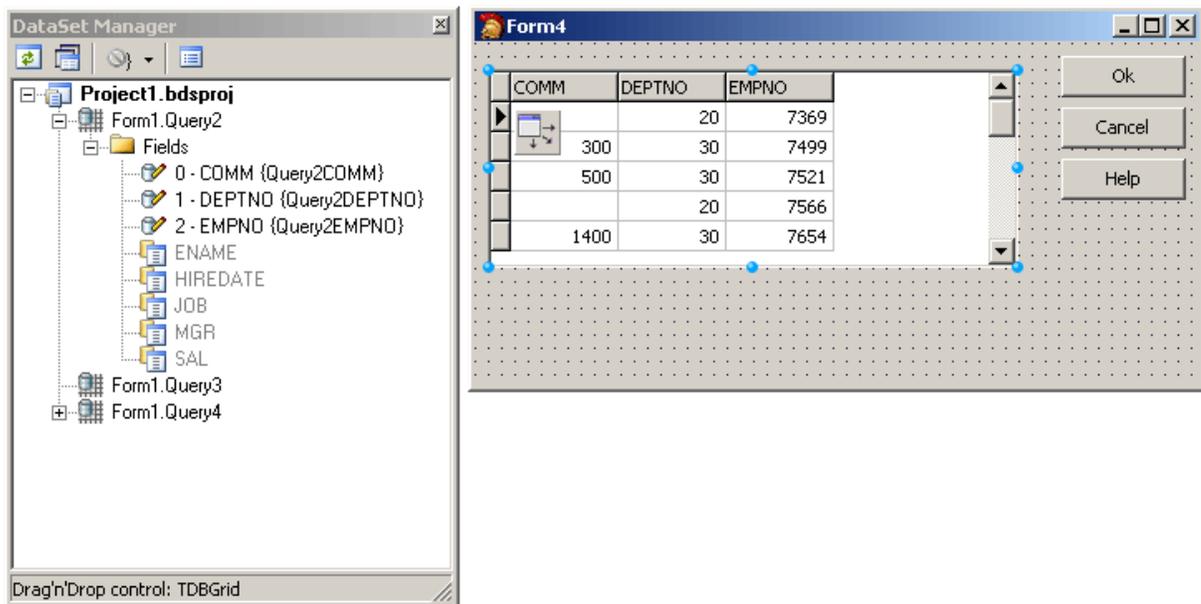


Creating Data-bound Controls

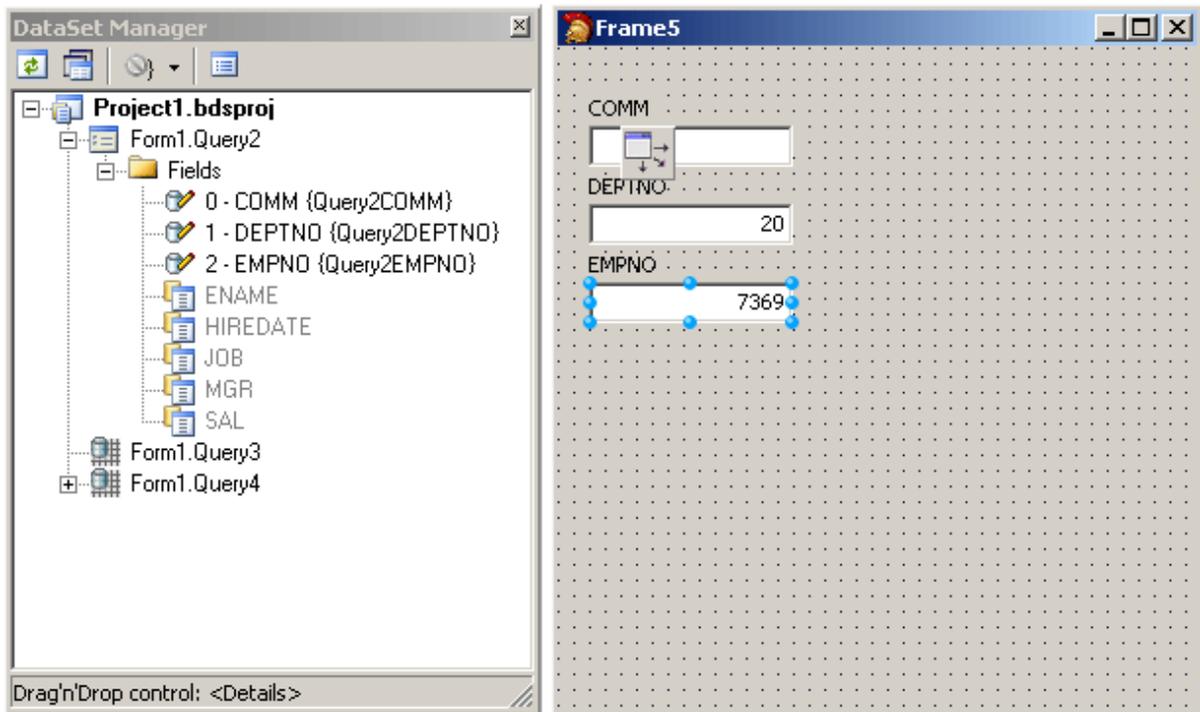
You can drag an item from the DataSet Manager window onto a form to create a new data-bound control. Each node in the DataSet Manager window allows you to choose the type of control that will be created when you drag it onto a form. You must choose between a Grid

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.

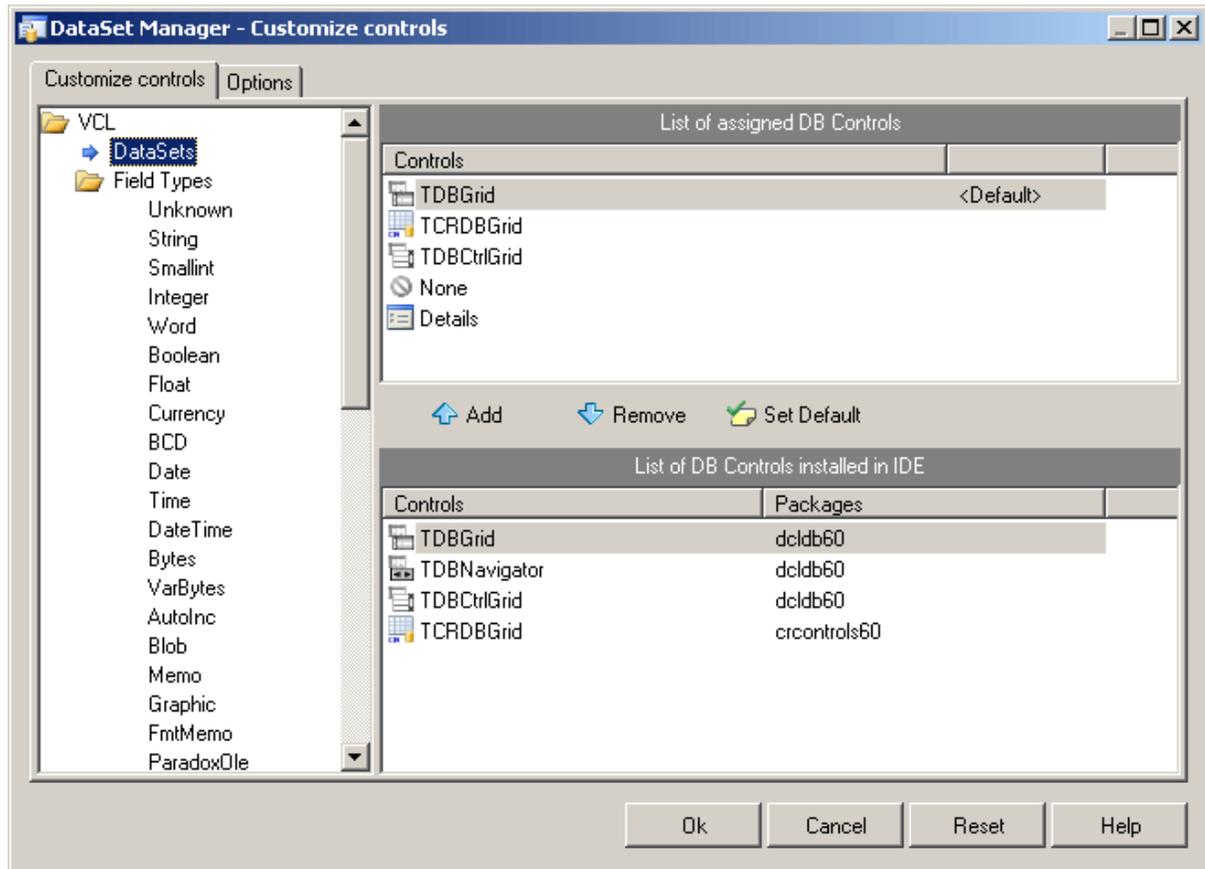


To use Details layout choose Details from the DataSet node drop-down control list in the DataSet Manager window. Then select required controls in the drop-down control list for each DataSet field. DataSet fields must be created. After setting required options you can drag the DataSet to the form from the DataSet wizard. 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 - Customize 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.

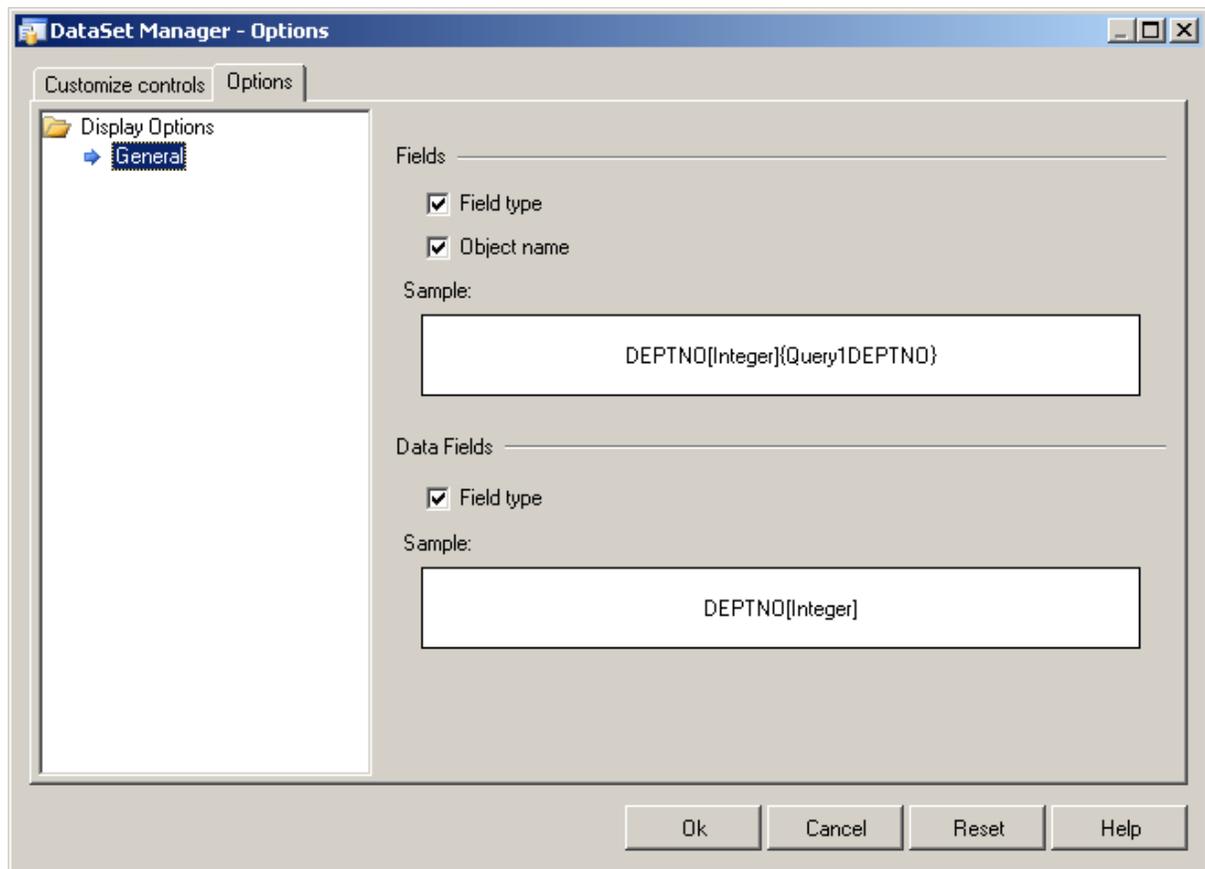
To create a field based on the database table column right-click the Fields node and select *Create Field* from the popup menu or press <Insert>. Note that after you add at least one field manually, DataSet fields corresponding to data fields will not be generated automatically when you drag the DataSet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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.13 Connection Pooling

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

connection process.

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

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

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

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

are broken.

The [PoolingOptions.MaxPoolSize](#) property limits the count of connections that can be active at the same time. If maximum count of connections is active and some TCustomDACConnection component tries to connect, it will have to wait until any of TCustomDACConnection components disconnect. Maximum wait time is 30 seconds. If active connections' count does not decrease during 30 seconds, the [TCustomDACConnection](#) 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 [TCustomDACConnection](#) 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 [TCustomDACConnection](#) component disconnects, the [RemoveFromPool](#) method of TCustomDACConnection can be used. You can also free all connection in the pool by using the class procedures Clear or AsyncClear of TIBConnectionPoolManager. These procedures can be useful when you know that all connections will be broken for some reason.

It is recommended to use connection pooling with the [DisconnectMode](#) option of the [TCustomDACConnection](#) component set to True. In this case internal connections can be shared between [TCustomDACConnection](#) components. When some operation is performed on the TCustomDACConnection 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 [TCustomDACConnection](#) component it can use the same connection from the pool.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.14 BLOB Data Types

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

IBDAC components provide following features for working with BLOBs:

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

IBDAC components support InterBase BLOB datatype. You can retrieve values of BLOB fields using TIBCQuery component the same way as you do for another fields.

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

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

Set [TCustomIBCDataSet.Options.CacheBlobs](#) to False to access streamed BLOB values on server side without caching BLOBs on the client side. Only requested portions of data are

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

IBDAC components provides easy usage of InterBase BLOB subtype conversion, using BLOB filters. It allows to set BLOB parameter buffer (BPB), that is needed whenever a filter will be used when writing to or reading from a BLOB. BPB contains source and target subtype and charset (for text BLOBs). These parameters are set in [TIBCBlob](#) component by setting [TIBCBlob.CharsetID](#), [TIBCBlob.ConversionCharsetID](#), [TIBCBlob.ConversionSubType](#), [TIBCBlob.SubType](#) properties. In retrieval operations, when you set them before reading BLOB, `SubType` and `Charset` properties are considered actual subtype and charset of database BLOB data. Application will get data converted to `ConversionSubType` subtype and `ConversionCharset` charset.

In upload operations, `SubType` and `Charset` properties mean actual subtype and charset of BLOB data in the application. `ConversionSubType` and `ConversionCharset` properties must contain desired subtype and charset to save BLOB to database with.

To avoid unwanted conversions do not set these properties, or make sure that `Charset` equals to `ConversionCharset` and `SubType` equals to `ConversionSubType`.

Note that if there is no filter for pair of source and target subtype, no conversion is provided and BLOB data remains unconverted.

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

Use `P:Devart.Dac.TDADatasetOptions.CompressBlobMode` for managing BLOB compression. BLOBs can be stored compressed on the client side, on the server side (in database) or on the both sides. By default it has value `cbNone`, that means no compression is provided. Use `cbClient` value to store compressed blobs on client side. This saves client memory. BLOB data is stored unchanged in database, other application can read these BLOBs as usual. If `cbServer` value is used, BLOB data is stored compressed in database. It's decompressed on the client side. This saves server disk space and network traffic. Other

application can't process compressed BLOB data as usual. To use compressed BLOB data both on the client and server side use `cbClientServer` value. To use `cbClient`, `cbServer`, `cbClientServer` and `cbNone` constants you should add `MemData` unit to uses clause.

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

```

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

```

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

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

See Also

- [TIBCBlob](#)
- [TCompressedBlob](#)
- [TCustomDADataset.Options](#)
- [TCustomIBCDataset.Options](#)
- [TDAParam.ParamType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.15 Unicode Character Data

Symbolic information in InterBase can be retrieved for the user as a different character

encoding according to the query. InterBase supports number of encoding formats including Unicode. IBDAC components support UTF-8 Unicode (Unicode_FSS) encoding formats for data fields.

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

Suppose that SIMPLE_TYPES table is created as:

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

Suppose we open the following SELECT statement in a dataset

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

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

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

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

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

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

To use Unicode values as parameters, previously you need to set a value of data type field to ftWideString or ftFixedWideChar for the fields of VARCHAR or CHAR types accordingly.

Otherwise after the execution of AsWideString or AsString operation data type field will be ftString by default.

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

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

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

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

See Also

- [TIBCConnection.Options](#)
- [TCustomIBCDataSet.Options](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.16 ARRAY Data Type

Some problems appear when you need to use large arrays in dataset. As IBDAC creates one field for each array item great number of TField objects is created. As a result of it the performance decreases. So IBDAC has the limitation and creates fields for first 1000 items. However, you can access all items with TIBCArray object.

If such table is created

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

If `ComplexArrayFields` is `False` you can access array item using `TIBCArrayField`

```
Value := TIBCArrayField(IBCQuery1.FieldByName('CHAR_ARRAY')).AsArray.GetItem
```

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

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

If `ObjectField` property is `True` this code is right

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

Using `TIBCDataSet.GetArray` you can access to array items through `TIBCArray` object

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

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

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

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

For example:

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

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.17 TIBCQuery Component

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

See Also

- [TIBCQuery](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.18 DBMonitor

To extend monitoring capabilities of IBDAC applications there is an additional tool called DBMonitor. It is provided as an alternative to Borland SQL Monitor which is also supported by IBDAC.

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 [TIBCSQLMonitor](#) 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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.19 Writing GUI Applications with IBDAC

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

Delphi and C++Builder

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.20 Compatibility with Previous Versions

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

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

DBAccess.SQLGeneratorCompatibility:

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

MemDS.SendDataSetChangeEventAfterOpen:

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

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

MemDS.DoNotRaiseExcetionOnUaFail:

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

To restore the old behaviour, set DoNotRaiseExcetionOnUaFail to True.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.21 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.

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:

TYPE	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.

TYPE	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:

TYPE	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:

TYPE	32-bit	64-bit
NativeInt	4 bytes	8 bytes
NativeUInt	4 bytes	8 bytes

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

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

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

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

Out parameters

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

For example:

```
procedure MyProc(out value: NativeInt);
begin
```

```

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

```

Win API

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

Correct:

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

Wrong:

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

Replace SetWindowLong/GetWindowLog 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 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

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.zip 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/Tokyo/en/Running_the_Platform_Assistant_on_Windows

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

4.22 Database Specific Aspects of 64-bit Development

InterBase and FireBird Connectivity Aspects

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5 Reference

This page shortly describes units that exist in IBDAC.

Units

Unit Name	Description
CRAccess	This unit contains base classes for accessing databases.
CRBatchMove	This unit contains implementation of the TCRBatchMove component.
CREncryption	This unit contains base classes for data encryption.

CRGrid	This unit contains the TCRDBGrid component.
CRVio	This unit contains the TIPVersion enumeration.
DAAlerter	This unit contains the base class for the TIBCAlerter component.
DADump	This unit contains the base class for the TIBCDump component.
DALoader	This unit contains the base class for the TIBCLoader component.
DAScript	This unit contains the base class for the TIBCScript component.
DASQLMonitor	This unit contains the base class for the TIBCSQLMonitor component.
DBAccess	This unit contains base classes for most of the components.
Devart.Dac.DataAdapter	This unit contains implementation of the DADDataAdapter class.
Devart.IbDac.DataAdapter	This unit contains implementation of the IBIDataAdapter class.
IBC	This unit contains main components of IBDAC.
IBCAdmin	This unit contains implementation of components, used for InterBase/Firebird server administration.
IBCAlerter	This unit contains implementation of the TIBCAlerter component.
IBCArray	Description is not available at the moment.
IBCClasses	IBCClasses unit defines the following data type constants: dtDbKey

	dtFixedChar dtFixedWideChar
IBCConnectionPool	This unit contains the TIBCConnectionPoolManager class for managing connection pool.
IBCDDataTypeMap	This unit contains the implementation of mapping between InterBase/Firebird and Delphi data types.
IBCErrors	IBCErrors unit implements the EIBCErrors class.
IBCLoader	This unit contains implementation of the TIBCLoader component.
IBCScript	This unit contains implementation of the TIBCScript component.
IBCSQLMonitor	This unit contains implementation of the TIBCSQLMonitor component.
IbDacVcl	This unit contains the visual constituent of IBDAC.
MemData	This unit contains classes for storing data in memory.
MemDS	This unit contains implementation of the TMemDataSet class.
VirtualDataSet	This unit contains implementation of the TVirtualDataSet component.
VirtualTable	This unit contains implementation of the TVirtualTable component.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.1 CRAccess

This unit contains base classes for accessing databases.

Classes

Name	Description
TCRCursor	A base class for classes that work with database cursors.

Types

Name	Description
TBeforeFetchProc	This type is used for the TCustomDADataset.BeforeFetch event.

Enumerations

Name	Description
TCRIsolationLevel	Specifies how to handle transactions containing database modifications.
TCRTransactionAction	Specifies the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.
TCursorState	Used to set cursor state

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.1.1 Classes

Classes in the **CRAccess** unit.

Classes

Name	Description
TCRCursor	A base class for classes that work with database cursors.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.1.1.1 TCRCursor Class

A base class for classes that work with database cursors.

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

Unit

[CRAccess](#)

Syntax

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

Remarks

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

Inheritance Hierarchy

[TSharedObject](#)

TCRCursor

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.1.1.1.1 Members

[TCRCursor](#) class overview.

Properties

Name	Description
RefCount (inherited from TSharedObject)	Used to return the count of reference to a TSharedObject object.

Methods

Name	Description
AddRef (inherited from TSharedObject)	Increments the reference count for the number of references dependent on the TSharedObject object.

Release (inherited from TSharedObject)	Decrements the reference count.
---	---------------------------------

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.1.2 Types

Types in the **CRAccess** unit.

Types

Name	Description
TBeforeFetchProc	This type is used for the TCustomDADDataSet.BeforeFetch event.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.1.2.1 TBeforeFetchProc Procedure Reference

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

Unit

[CRAccess](#)

Syntax

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

Parameters

Cancel

True, if the current fetch operation should be aborted.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.1.3 Enumerations

Enumerations in the **CRAccess** unit.

Enumerations

Name	Description
TCRIsolationLevel	Specifies how to handle transactions containing database modifications.
TCRTransactionAction	Specifies the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.
TCursorState	Used to set cursor state

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.1.3.1 TCRIsolationLevel Enumeration

Specifies how to handle transactions containing database modifications.

Unit

[CRAccess](#)

Syntax

```
TCRIsolationLevel = (ilReadCommitted, ilReadUnCommitted,
ilRepeatableRead, ilIsolated, ilSnapshot, ilCustom);
```

Values

Value	Meaning
ilCustom	The parameters of the transaction are set manually in the Params property.
ilIsolated	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 the concurrent accesses to the

	same set of data, but may lead to a greater latency at times of a congested database environment.
ilReadCommitted	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 the isolation level. The default value.
ilReadUnCommitted	The most unrestricted level of the transaction isolation. All types of data access interferences are possible. Mainly used for browsing database and to receive instant data with prospective changes.
ilRepeatableRead	Prevents concurrent transactions from modifying data in the 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).
ilSnapshot	Uses row versioning. Provides transaction-level read consistency. A data snapshot is taken when the snapshot transaction starts, and remains consistent for the duration of a transaction.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.1.3.2 TCRTransactionAction Enumeration

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

Unit

[CRAccess](#)

Syntax

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

Values

Value	Meaning
taCommit	Transaction is committed.
taRollback	Transaction is rolled back.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.1.3.3 TCursorState Enumeration

Used to set cursor state

Unit

[CRAccess](#)

Syntax

```
TCursorState = (csInactive, csOpen, csParsed, csPrepared, csBound,
csExecuteFetchAll, csExecuting, csExecuted, csFetching,
csFetchingAll, csFetched);
```

Values

Value	Meaning
csBound	Parameters bound
csExecuted	Statement successfully executed
csExecuteFetchAll	Set before FetchAll
csExecuting	Statement is set before executing
csFetched	Fetch finished or canceled
csFetching	Set on first
csFetchingAll	Set on the FetchAll start
csInactive	Default state
csOpen	statement open
csParsed	Statement parsed
csPrepared	Statement prepared

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2 CRBatchMove

This unit contains implementation of the TCRBatchMove component.

Classes

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

TCRBatchMove	Transfers records between datasets.
------------------------------	-------------------------------------

Types

Name	Description
TCRBatchMoveProgressEvent	This type is used for the TCRBatchMove.OnBatchMoveProgress event.

Enumerations

Name	Description
TCRBatchMode	Used to set the type of the batch operation that will be executed after calling the TCRBatchMove.Execute method.
TCRFieldMappingMode	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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1 Classes

Classes in the **CRBatchMove** unit.

Classes

Name	Description
TCRBatchMove	Transfers records between datasets.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1 TCRBatchMove Class

Transfers records between datasets.

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

Unit

[CRBatchMove](#)

Syntax

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

Remarks

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

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.1 Members

[TCRBatchMove](#) class overview.

Properties

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2 Properties

Properties of the **TCRBatchMove** class.

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

Public

Name	Description
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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.1 AbortOnKeyViol Property

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

Class

[TCRBatchMove](#)

Syntax

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

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.2 AbortOnProblem Property

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

Class

[TCRBatchMove](#)

Syntax

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

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.3 ChangedCount Property

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

Class

[TCRBatchMove](#)

Syntax

```
property ChangedCount: Integer;
```

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.4 CommitCount Property

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

Class

[TCRBatchMove](#)

Syntax

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

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.5 Destination Property

Used to specify the destination dataset for the batch operation.

Class

[TCRBatchMove](#)

Syntax

```
property Destination: TDataSet;
```

Remarks

Specifies the destination dataset for the batch operation.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.6 FieldMappingMode Property

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

Class

[TCRBatchMove](#)

Syntax

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

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.7 KeyViolCount Property

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

Class

[TCRBatchMove](#)

Syntax

```
property KeyViolCount: Integer;
```

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.8 Mappings Property

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

Class

[TCRBatchMove](#)

Syntax

```
property Mappings: TStrings;
```

Remarks

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

ColName

Example

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

```
DestColName=SourceColName
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.9 Mode Property

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

Class

[TCRBatchMove](#)

Syntax

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

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.10 MovedCount Property

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

Class

[TCRBatchMove](#)

Syntax

```
property MovedCount: Integer;
```

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.11 ProblemCount Property

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

Class

[TCRBatchMove](#)

Syntax

```
property ProblemCount: Integer;
```

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.12 RecordCount Property

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

Class

[TCRBatchMove](#)

Syntax

```
property RecordCount: Integer 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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.2.13 Source Property

Used to specify the source dataset for the batch operation.

Class

[TCRBatchMove](#)

Syntax

```
property Source: TDataSet;
```

Remarks

Specifies the source dataset for the batch operation.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.3 Methods

Methods of the **TCRBatchMove** class.

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

Public

Name	Description
Execute	Performs the batch operation.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.3.1 Execute Method

Performs the batch operation.

Class

[TCRBatchMove](#)

Syntax

```
procedure Execute;
```

Remarks

Call the Execute method to perform the batch operation.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.2.1.1.4 Events

Events of the **TCRBatchMove** class.

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

Published

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

See Also

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

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.1.1.4.1 OnBatchMoveProgress Event

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

Class

[TCRBatchMove](#)

Syntax

```
property OnBatchMoveProgress: TCRBatchMoveProgressEvent;
```

Remarks

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

© 1997-2024

Devart. All Rights

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.2.2 Types

Types in the **CRBatchMove** unit.

Types

Name	Description
TCRBatchMoveProgressEvent	This type is used for the TCRBatchMove.OnBatchMoveProgress event.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.2.1 TCRBatchMoveProgressEvent Procedure Reference

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

Unit

[CRBatchMove](#)

Syntax

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

Parameters

Sender

An object that raised the event.

Percent

Percentage of the batch operation progress.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.3 Enumerations

Enumerations in the **CRBatchMove** unit.

Enumerations

Name	Description
TCRBatchMode	Used to set the type of the batch operation that will be executed after calling the TCRBatchMove.Execute method.
TCRFieldMappingMode	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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.2.3.1 TCRBatchMode Enumeration

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

Unit

[CRBatchMove](#)

Syntax

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

Values

Value	Meaning
bmAppend	Appends the records from the source dataset to the destination dataset. The default mode.
bmAppendUpdate	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.
bmDelete	Deletes records from the destination dataset if there are matching records in the source dataset.
bmUpdate	Replaces records in the destination dataset with the matching records from the source dataset.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.2.3.2 TCRFieldMappingMode Enumeration

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

Unit

[CRBatchMove](#)

Syntax

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

Values

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3 CREncryption

This unit contains base classes for data encryption.

Classes

Name	Description
TCREncryptor	The class that performs data encryption and decryption in a client application using various encryption algorithms .

Enumerations

Name	Description
TCREncDataHeader	Specifies whether the additional information is

	stored with the encrypted data.
TCREncryptionAlgorithm	Specifies the algorithm of data encryption.
TCRHashAlgorithm	Specifies the algorithm of generating hash data.
TCRInvalidHashAction	Specifies the action to perform on data fetching when hash data is invalid.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.1 Classes

Classes in the **CREncryption** unit.

Classes

Name	Description
TCREncryptor	The class that performs data encryption and decryption in a client application using various encryption algorithms .

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.1.1 TCREncryptor Class

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

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

Unit

[CREncryption](#)

Syntax

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.3.1.1.1 Members

[TCREncryptor](#) class overview.

Properties

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.

Methods

Name	Description
SetKey	Sets a key, using which data is encrypted.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.3.1.1.2 Properties

Properties of the **TCREncryptor** class.

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

Published

Name	Description
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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.1.1.2.1 DataHeader Property

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

Class

[TCREncryptor](#)

Syntax

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

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.1.1.2.2 EncryptionAlgorithm Property

Specifies the algorithm of data encryption.

Class

[TCREncryptor](#)

Syntax

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

Remarks

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

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.1.1.2.3 HashAlgorithm Property

Specifies the algorithm of generating hash data.

Class

[TCREncryptor](#)

Syntax

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

Remarks

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

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.1.1.2.4 InvalidHashAction Property

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

Class

[TCREncryptor](#)

Syntax

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

Remarks

Use InvalidHashAction to specify the action to perform on data fetching when hash data is invalid. This property is used only if hash is stored with the encrypted data (the [DataHeader](#) property is set to [ehTagAndHash](#)). Default value is [ihFail](#).

If the DataHeader property is set to ehTagAndHash, then on data fetching from a server the hash check is performed for each record. After data decryption its hash is calculated and compared with the hash stored in the field. If these values don't coincide, it means that the stored data is incorrect, and depending on the value of the InvalidHashAction property one of the following actions is performed:

[ihFail](#) - the EInvalidHash exception is raised and further data reading from the server is interrupted.

[ihSkipData](#) - the value of the field for this record is set to Null. No exception is raised.

[ihIgnoreError](#) - in spite of the fact that the data is not valid, the value is set in the field. No exception is raised.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.1.1.2.5 Password Property

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

Class

[TCREncryptor](#)

Syntax

```
property Password: string stored False;
```

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.1.1.3 Methods

Methods of the **TCREncryptor** class.

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

Public

Name	Description
SetKey	Sets a key, using which data is encrypted.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.1.1.3.1 SetKey Method

Sets a key, using which data is encrypted.

Class

[TCREncryptor](#)

Syntax

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

Parameters

Key

Holds bytes that represent a key.

Offset

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

Count

Number of bytes to use from Key.

Remarks

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

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.2 Enumerations

Enumerations in the **CREncryption** unit.

Enumerations

Name	Description
TCREncDataHeader	Specifies whether the additional information is stored with the encrypted data.
TCREncryptionAlgorithm	Specifies the algorithm of data encryption.
TCRHashAlgorithm	Specifies the algorithm of generating hash data.
TCRInvalidHashAction	Specifies the action to perform on data fetching when hash data is invalid.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.2.1 TCREncDataHeader Enumeration

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

Unit

[CREncryption](#)

Syntax

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

Values

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.2.2 TCREncryptionAlgorithm Enumeration

Specifies the algorithm of data encryption.

Unit

[CREncryption](#)

Syntax

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

Values

Value	Meaning
eaAES128	The AES encryption algorithm with key size of 128 bits is used.
eaAES192	The AES encryption algorithm with key size of 192 bits is used.
eaAES256	The AES encryption algorithm with key size of 256 bits is used.

eaBlowfish	The Blowfish encryption algorithm is used.
eaCast128	The CAST-128 encryption algorithm with key size of 128 bits is used.
eaRC4	The RC4 encryption algorithm is used.
eaTripleDES	The Triple DES encryption algorithm is used.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.2.3 TCRHashAlgorithm Enumeration

Specifies the algorithm of generating hash data.

Unit

[CREncryption](#)

Syntax

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

Values

Value	Meaning
haMD5	The MD5 hash algorithm is used.
haSHA1	The SHA-1 hash algorithm is used.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.3.2.4 TCRInvalidHashAction Enumeration

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

Unit

[CREncryption](#)

Syntax

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

Values

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.4 CRVio

This unit contains the TIPVersion enumeration.

Enumerations

Name	Description
TIPVersion	Specifies Internet Protocol version.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.4.1 Enumerations

Enumerations in the **CRVio** unit.

Enumerations

Name	Description
TIPVersion	Specifies Internet Protocol version.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.4.1.1 TIPVersion Enumeration

Specifies Internet Protocol version.

Unit

[CRVio](#)

Syntax

```
TIPVersion = (ivIPv4, ivIPv6, ivIPBoth);
```

Values

Value	Meaning
ivIPBoth	Specifies that either IPv6 or IPv4 Internet Protocol version is used
ivIPv4	Specifies that the IPv4 Internet Protocol version is used
ivIPv6	Specifies that the IPv6 Internet Protocol version is used

Remarks

Note: When the TIPVersion property is set to **ivIPBoth**, a connection attempt is made via IPv6 if it is enabled in the operating system settings. If the connection attempt fails, a new connection attempt is made via IPv4.

See Also

- [TIBCConnectionOptions.IPVersion](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5 DAAlerter

This unit contains the base class for the TIBCAAlerter component.

Classes

Name	Description
TDAAAlerter	A base class that defines functionality for database event notification.

Types

Name	Description
TAlerterErrorEvent	This type is used for the TDAAAlerter.OnError event.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.1 Classes

Classes in the **DAALerter** unit.

Classes

Name	Description
TDAALerter	A base class that defines functionality for database event notification.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.1.1 TDAALerter Class

A base class that defines functionality for database event notification.

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

Unit

[DAALerter](#)

Syntax

```
TDAALerter = class(TComponent);
```

Remarks

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

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

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.1.1.1 Members

[TDAAlerter](#) class overview.

Properties

Name	Description
Active	Used to determine if TDAAlerter waits for messages.
AutoRegister	Used to automatically register events whenever connection opens.
Connection	Used to specify the connection for TDAAlerter.

Methods

Name	Description
SendEvent	Sends an event with Name and content Message.
Start	Starts waiting process.
Stop	Stops waiting process.

Events

Name	Description
OnError	Occurs if an exception occurs in waiting process

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.1.1.2 Properties

Properties of the **TDAAlerter** class.

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

Public

Name	Description
Active	Used to determine if TDAAlerter waits for messages.
AutoRegister	Used to automatically register events whenever connection opens.
Connection	Used to specify the connection for TDAAlerter.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.5.1.1.2.1 Active Property

Used to determine if TDAAlerter waits for messages.

Class

[TDAAlerter](#)

Syntax

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

Remarks

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

See Also

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

© 1997-2024

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

Devart. All Rights Reserved.

5.5.1.1.2.2 AutoRegister Property

Used to automatically register events whenever connection opens.

Class

[TDAAlerter](#)

Syntax

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

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.1.1.2.3 Connection Property

Used to specify the connection for TDAAlerter.

Class

[TDAAlerter](#)

Syntax

```
property Connection: TCustomDAConnection;
```

Remarks

Use the Connection property to specify the connection for TDAAlerter.

See Also

- [TIBCCConnection](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.1.1.3 Methods

Methods of the **TDAAlert** class.

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

Public

Name	Description
SendEvent	Sends an event with Name and content Message.
Start	Starts waiting process.
Stop	Stops waiting process.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.1.1.3.1 SendEvent Method

Sends an event with Name and content Message.

Class

[TDAAlert](#)

Syntax

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

Parameters

EventName

Holds the event name.

Message

Holds the content Message of the event.

Remarks

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

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.1.1.3.2 Start Method

Starts waiting process.

Class

[TDAAlerter](#)

Syntax

```
procedure Start;
```

Remarks

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

See Also

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

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.1.1.3.3 Stop Method

Stops waiting process.

Class

[TDAAlerter](#)

Syntax

```
procedure Stop;
```

Remarks

Call Stop method to end waiting process.

See Also

- [Start](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.1.1.4 Events

Events of the **TDAAlerter** class.

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

Public

Name	Description
OnError	Occurs if an exception occurs in waiting process

See Also

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

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.1.1.4.1 OnError Event

Occurs if an exception occurs in waiting process

Class

[TDAAlerter](#)

Syntax

```
property OnError: TAAlerterErrorEvent;
```

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.2 Types

Types in the **DAAlerter** unit.

Types

Name	Description
TAlerterErrorEvent	This type is used for the TDAAlerter.OnError event.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.5.2.1 TAlerterErrorEvent Procedure Reference

This type is used for the TDAAlerter.OnError event.

Unit

[DAAlerter](#)

Syntax

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

Parameters

Sender

An object that raised the event.

E

Exception object.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6 DADump

This unit contains the base class for the TIBCDump component.

Classes

Name	Description
TDADump	A base class that defines functionality for descendant classes that dump database objects to a script.
TDADumpOptions	This class allows setting up the behaviour of the TDADump class.

Types

Name	Description
TDABackupProgressEvent	This type is used for the TDADump.OnBackupProgress event.
TDARestoreProgressEvent	This type is used for the TDADump.OnRestoreProgress event.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1 Classes

Classes in the **DADump** unit.

Classes

Name	Description
TDADump	A base class that defines functionality for descendant classes that dump database objects to a script.
TDADumpOptions	This class allows setting up the behaviour of the TDADump class.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1 TDADump Class

A base class that defines functionality for descendant classes that dump database objects to a script.

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

Unit

[DADump](#)

Syntax

```
TDADump = class(TComponent);
```

Remarks

TDADump is a base class that defines functionality for descendant classes that dump database objects to a script. Applications never use TDADump objects directly. Instead they use descendants of TDADump.

Use TDADump descendants to dump database objects, such as tables, stored procedures, and functions for backup or for transferring the data to another SQL server. The dump contains SQL statements to create the table or other database objects and/or populate the table.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.1 Members

[TDADump](#) class overview.

Properties

Name	Description
Connection	Used to specify a connection object that will be used to connect to a data store.

Debug	Used to display the statement that is being executed and the values and types of its parameters.
Options	Used to specify the behaviour of a TDADump component.
SQL	Used to set or get the dump script.
TableNames	Used to set the names of the tables to dump.

Methods

Name	Description
Backup	Dumps database objects to the TDADump.SQL 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
OnBackupProgress	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	Occurs when InterBase raises some error on TDADump.Restore .
OnRestoreProgress	Occurs to indicate the TDADump.Restore , TDADump.RestoreFromFile , or TDADump.RestoreFromStream method execution progress.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.2 Properties

Properties of the **TDADump** class.

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

Public

Name	Description
Connection	Used to specify a connection object that will be used to connect to a data store.
Options	Used to specify the behaviour of a TDADump component.

Published

Name	Description
Debug	Used to display the statement that is being executed and the values and types of its parameters.
SQL	Used to set or get the dump script.
TableNames	Used to set the names of the tables to dump.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.2.1 Connection Property

Used to specify a connection object that will be used to connect to a data store.

Class

[TDADump](#)

Syntax

```
property Connection: TCustomDACConnection;
```

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 TCustomDACConnection or its descendant class objects.

At runtime, link an instance of a TCustomDACConnection descendant to the Connection property.

See Also

- [TCustomDACConnection](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.2.2 Debug Property

Used to display the statement that is being executed and the values and types of its parameters.

Class

[TDADump](#)

Syntax

```
property Debug: boolean default False;
```

Remarks

Set the Debug property to True to display the statement that is being executed and the values and types of its parameters.

You should add the IdacVcl unit to the uses clause of any unit in your project to make the Debug property work.

Note: If TIBCSQLMonitor is used in the project and the TIBCSQLMonitor.Active property is set to False, the debug window is not displayed.

See Also

- [TCustomDADDataSet.Debug](#)
- [TCustomDASQL.Debug](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.2.3 Options Property

Used to specify the behaviour of a TDADump component.

Class

[TDADump](#)

Syntax

```
property Options: TDADumpOptions;
```

Remarks

Use the Options property to specify the behaviour of a TDADump component.

Descriptions of all options are in the table below.

Option Name	Description
AddDrop	Used to add drop statements to a script before creating statements.
CompleteInsert	Used to explicitly specify the table fields names when generating the INSERT SQL query. The default value is False.
GenerateHeader	Used to add a comment header to a script.
QuoteNames	Used for TDADump to quote all database object names in generated SQL statements.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.2.4 SQL Property

Used to set or get the dump script.

Class

[TDADump](#)

Syntax

```
property SQL: TStrings;
```

Remarks

Use the SQL property to get or set the dump script. The SQL property stores script that is executed by the [Restore](#) method. This property will store the result of [Backup](#) and [BackupQuery](#). At design time the SQL property can be edited by invoking the String List editor in Object Inspector.

See Also

- [Restore](#)
- [Backup](#)
- [BackupQuery](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.2.5 TableNames Property

Used to set the names of the tables to dump.

Class

[TDADump](#)

Syntax

```
property TableNames: string;
```

Remarks

Use the TableNames property to set the names of the tables to dump. Table names must be separated with semicolons. If the property is empty, the [Backup](#) method will dump all available tables.

See Also

- [Backup](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.3 Methods

Methods of the **TDADump** class.

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

Public

Name	Description
Backup	Dumps database objects to the TDADump.SQL 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.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.3.1 Backup Method

Dumps database objects to the [SQL](#) property.

Class

[TDADump](#)

Syntax

```
procedure Backup;
```

Remarks

Call the Backup method to dump database objects. The result script will be stored in the [SQL](#) property.

See Also

- [SQL](#)
- [Restore](#)
- [BackupToFile](#)
- [BackupToStream](#)
- [BackupQuery](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.6.1.1.3.2 BackupQuery Method

Dumps the results of a particular query.

Class

[TDADump](#)

Syntax

```
procedure BackupQuery(const Query: string);
```

Parameters

Query

Holds a query used for data selection.

Remarks

Call the BackupQuery method to dump the results of a particular query. Query must be a valid select statement. If this query selects data from several tables, only data of the first table in the from list will be dumped.

See Also

- [Restore](#)
- [Backup](#)
- [BackupToFile](#)
- [BackupToStream](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.3.3 BackupToFile Method

Dumps database objects to the specified file.

Class

[TDADump](#)

Syntax

```
procedure BackupToFile(const FileName: string; const Query:  
string = '');
```

Parameters

FileName

Holds the file name to dump database objects to.

Query

Your query to receive the data for dumping.

Remarks

Call the BackupToFile method to dump database objects to the specified file.

See Also

- [RestoreFromStream](#)
- [Backup](#)
- [BackupToStream](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.3.4 BackupToStream Method

Dumps database objects to the stream.

Class

[TDADump](#)

Syntax

```
procedure BackupToStream(Stream: TStream; const Query: string =  
'');
```

Parameters

Stream

Holds the stream to dump database objects to.

Query

Your query to receive the data for dumping.

Remarks

Call the BackupToStream method to dump database objects to the stream.

See Also

- [RestoreFromStream](#)
- [Backup](#)
- [BackupToFile](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.3.5 Restore Method

Executes a script contained in the SQL property.

Class

[TDADump](#)

Syntax

```
procedure Restore;
```

Remarks

Call the Restore method to execute a script contained in the SQL property.

See Also

- [RestoreFromFile](#)
- [RestoreFromStream](#)
- [Backup](#)
- [SQL](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.3.6 RestoreFromFile Method

Executes a script from a file.

Class

[TDADump](#)

Syntax

```
procedure RestoreFromFile(const FileName: string);  
overload; procedure RestoreFromFile(const FileName: string;  
Encoding: TEncoding); overload;
```

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](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.3.7 RestoreFromStream Method

Executes a script received from the stream.

Class

[TDADump](#)

Syntax

```
procedure RestoreFromStream(Stream: TStream);
```

Parameters

Stream

Holds a stream to receive a script to be executed.

Remarks

Call the `RestoreFromStream` method to execute a script received from the stream.

See Also

- [Restore](#)
- [RestoreFromFile](#)
- [BackupToStream](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.4 Events

Events of the **TDADump** class.

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

Published

Name	Description
OnBackupProgress	Occurs to indicate the TDADump.Backup , <code>M:Devart.Dac.TDADump.BackupToFile(System.String)</code> or <code>M:Devart.Dac.TDADump.BackupToStream(Borland.Vcl.TStream)</code> method execution progress.
OnError	Occurs when InterBase raises some error on TDADump.Restore .
OnRestoreProgress	Occurs to indicate the TDADump.Restore , TDADump.RestoreFromFile , or TDADump.RestoreFromStream method execution progress.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.4.1 OnBackupProgress Event

Occurs to indicate the [Backup](#), M:Devart.Dac.TDADump.BackupToFile(System.String) or M:Devart.Dac.TDADump.BackupToStream(Borland.Vcl.TStream) method execution progress.

Class

[TDADump](#)

Syntax

```
property OnBackupProgress: TDABackupProgressEvent;
```

Remarks

The OnBackupProgress event occurs several times during the dumping process of the [Backup](#), M:Devart.Dac.TDADump.BackupToFile(System.String), or M:Devart.Dac.TDADump.BackupToStream(Borland.Vcl.TStream) method execution and indicates its progress. ObjectName parameter indicates the name of the currently dumping database object. ObjectNum shows the number of the current database object in the backup queue starting from zero. ObjectCount shows the quantity of database objects to dump. Percent parameter shows the current percentage of the current table data dumped, not the current percentage of the entire dump process.

See Also

- [Backup](#)
- [BackupToFile](#)
- [BackupToStream](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.6.1.1.4.2 OnError Event

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

Class

[TDADump](#)

Syntax

```
property OnError: TOnErrorEvent;
```

Remarks

The OnError event occurs when InterBase raises some error on [Restore](#).

Action indicates the action to take when the OnError handler exits. On entry into the handler, Action is always set to eaException.

Note: You should add the DAScript module to the 'uses' list to use the OnError event handler.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.1.4.3 OnRestoreProgress Event

Occurs to indicate the [Restore](#), [RestoreFromFile](#), or [RestoreFromStream](#) method execution progress.

Class

[TDADump](#)

Syntax

```
property OnRestoreProgress: TDARestoreProgressEvent;
```

Remarks

The OnRestoreProgress event occurs several times during the dumping process of the [Restore](#), [RestoreFromFile](#), or [RestoreFromStream](#) method execution and indicates its progress. The Percent parameter of the OnRestoreProgress event handler indicates the

percentage of the whole restore script execution.

See Also

- [Restore](#)
- [RestoreFromFile](#)
- [RestoreFromStream](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.2 TDADumpOptions Class

This class allows setting up the behaviour of the TDADump class.

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

Unit

[DADump](#)

Syntax

```
TDADumpOptions = class(TPersistent);
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.2.1 Members

[TDADumpOptions](#) class overview.

Properties

Name	Description
AddDrop	Used to add drop statements to a script before creating statements.
CompleteInsert	Used to explicitly specify the table fields names when generating the INSERT SQL query. The default value is False.

GenerateHeader	Used to add a comment header to a script.
QuoteNames	Used for TDADump to quote all database object names in generated SQL statements.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.2.2 Properties

Properties of the **TDADumpOptions** class.

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

Published

Name	Description
AddDrop	Used to add drop statements to a script before creating statements.
CompleteInsert	Used to explicitly specify the table fields names when generating the INSERT SQL query. The default value is False.
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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.2.2.1 AddDrop Property

Used to add drop statements to a script before creating statements.

Class

[TDADumpOptions](#)

Syntax

```
property AddDrop: boolean default True;
```

Remarks

Use the AddDrop property to add drop statements to a script before creating statements.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.2.2.2 CompleteInsert Property

Used to explicitly specify the table fields names when generating the INSERT SQL query. The default value is False.

Class

[TDADumpOptions](#)

Syntax

```
property CompleteInsert: boolean default False;
```

Remarks

If the CompleteInsert property is set to True, SQL query will include the field names, for example:

```
INSERT INTO dept(deptno, dname, loc) VALUES ('10', 'ACCOUNTING', 'NEW YORK');
```

If False, it won't include the field names, for example:

```
INSERT INTO dept VALUES ('10', 'ACCOUNTING', 'NEW YORK');
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.2.2.3 GenerateHeader Property

Used to add a comment header to a script.

Class

[TDADumpOptions](#)

Syntax

```
property GenerateHeader: boolean default True;
```

Remarks

Use the GenerateHeader property to add a comment header to a script. It contains script generation date, DAC version, and some other information.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.1.2.2.4 QuoteNames Property

Used for TDADump to quote all database object names in generated SQL statements.

Class

[TDADumpOptions](#)

Syntax

```
property QuoteNames: boolean default False;
```

Remarks

If the QuoteNames property is True, TDADump quotes all database object names in generated SQL statements.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.2 Types

Types in the **DADump** unit.

Types

Name	Description
TDABackupProgressEvent	This type is used for the TDADump.OnBackupProgress event.
TDARestoreProgressEvent	This type is used for the TDADump.OnRestoreProgress event.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.6.2.1 TDABackupProgressEvent Procedure Reference

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

Unit

[DADump](#)

Syntax

```
TDABackupProgressEvent = procedure (Sender: TObject; ObjectName:
string; ObjectNum: integer; ObjectCount: integer; Percent:
integer) of object;
```

Parameters

Sender

An object that raised the event.

ObjectName

The name of the currently dumping database object.

ObjectNum

The number of the current database object in the backup queue starting from zero.

ObjectCount

The quantity of database objects to dump.

Percent

The current percentage of the current table data dumped.

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.6.2.2 TDARestoreProgressEvent Procedure Reference

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

Unit

[DADump](#)

Syntax

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

Parameters

Sender

An object that raised the event.

Percent

The percentage of the whole restore script execution.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7 DALoader

This unit contains the base class for the TIBCLoader component.

Classes

Name	Description
TDAColumn	Represents the attributes for column loading.
TDAColumns	Holds a collection of TDAColumn objects.
TDALoader	This class allows loading external data into database.
TDALoaderOptions	Allows loading external data into database.

Types

Name	Description
TDAPutDataEvent	This type is used for the TDALoader.OnPutData event.
TGetColumnDataEvent	This type is used for the TDALoader.OnGetColumnData event.
TLoaderProgressEvent	This type is used for the TDALoader.OnProgress event.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1 Classes

Classes in the **DALoader** unit.

Classes

Name	Description
TDAColumn	Represents the attributes for column loading.
TDAColumns	Holds a collection of TDAColumn objects.
TDALoader	This class allows loading external data into database.
TDALoaderOptions	Allows loading external data into database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.1 TDAColumn Class

Represents the attributes for column loading.

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

Unit

[DALoader](#)

Syntax

```
TDAColumn = class(TCollectionItem);
```

Remarks

Each [TDALoader](#) uses [TDAColumns](#) to maintain a collection of TDAColumn objects.

TDAColumn object represents the attributes for column loading. Every TDAColumn object corresponds to one of the table fields with the same name as its [TDAColumn.Name](#) property.

To create columns at design-time use the column editor of the [TDALoader](#) component.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.1.1 Members

[TDAColumn](#) class overview.

Properties

Name	Description
FieldType	Used to specify the types of values that will be loaded.
Name	Used to specify the field name of loading table.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.1.2 Properties

Properties of the **TDAColumn** class.

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

Published

Name	Description
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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.1.2.1 FieldType Property

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

Class

[TDAColumn](#)

Syntax

```
property FieldType: TFieldType default ftstring;
```

Remarks

Use the FieldType property to specify the types of values that will be loaded. Field types for columns may not match data types for the corresponding fields in the database table.

[TDALoader](#) will cast data values to the types of their fields.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.1.2.2 Name Property

Used to specify the field name of loading table.

Class

[TDAColumn](#)

Syntax

```
property Name: string;
```

Remarks

Each TDAColumn corresponds to one field of the loading table. Use the Name property to specify the name of this field.

See Also

- [FieldType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.2 TDAColumns Class

Holds a collection of [TDAColumn](#) objects.

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

Unit

[DALoader](#)

Syntax

```
TDAColumns = class(TOwnedCollection);
```

Remarks

Each TDAColumns holds a collection of [TDAColumn](#) objects. TDAColumns maintains an index of the columns in its Items array. The Count property contains the number of columns in the collection. At design-time, use the Columns editor to add, remove, or modify columns.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.2.1 Members

[TDAColumns](#) class overview.

Properties

Name	Description
Items	Used to access individual columns.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.2.2 Properties

Properties of the **TDAColumns** class.

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

Public

Name	Description
Items	Used to access individual columns.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.2.2.1 Items Property(Indexer)

Used to access individual columns.

Class

[TDAColumns](#)

Syntax

```
property Items[Index: integer]: TDAColumn; default;
```

Parameters

Index

Holds the Index of [TDAColumn](#) to refer to.

Remarks

Use the Items property to access individual columns. The value of the Index parameter corresponds to the Index property of [TDAColumn](#).

See Also

- [TDAColumn](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3 TDALoader Class

This class allows loading external data into database.

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

Unit

[DALoader](#)

Syntax

```
TDALoader = class(TComponent);
```

Remarks

TDALoader allows loading external data into database. To specify the name of loading table set the [TDALoader.TableName](#) property. Use the [TDALoader.Columns](#) property to access individual columns. Write the [TDALoader.OnGetColumnData](#) or [TDALoader.OnPutData](#) event handlers to read external data and pass it to the database. Call the [TDALoader.Load](#) method to start loading data.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.1 Members

[TDALoader](#) class overview.

Properties

Name	Description
Columns	Used to add a TDAColumn object for each field that will be loaded.
Connection	property. Used to specify TCustomDAConnection in which TDALoader will be executed.
TableName	Used to specify the name of the table to which data will be loaded.

Methods

Name	Description
CreateColumns	Creates TDAColumn objects for all fields of the table with the same name as TDALoader.TableName .
Load	Starts loading data.
LoadFromDataSet	Loads data from the specified dataset.
PutColumnData	Overloaded. Puts the value of individual columns.

Events

Name	Description
OnGetColumnData	Occurs when it is needed to put column values.
OnProgress	Occurs if handling data loading progress of the TDALoader.LoadFromDataSet method is needed.
OnPutData	Occurs when putting loading data by rows is needed.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.2 Properties

Properties of the **TDALoader** class.

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

Public

Name	Description
Columns	Used to add a TDAColumn object for each field that will be loaded.
Connection	property. Used to specify TCustomDACConnection in which TDALoader will be executed.
TableName	Used to specify the name of the table to which data will be loaded.

See Also

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

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.2.1 Columns Property

Used to add a [TDAColumn](#) object for each field that will be loaded.

Class

[TDALoader](#)

Syntax

```
property columns: TDAColumns stored IsColumnsStored;
```

Remarks

Use the Columns property to add a [TDAColumn](#) object for each field that will be loaded.

See Also

- [TDAColumns](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.2.2 Connection Property

property. Used to specify TCustomDACConnection in which TDALoader will be executed.

Class

[TDALoader](#)

Syntax

```
property Connection: TCustomDACConnection;
```

Remarks

Use the Connection property to specify TCustomDACConnection in which TDALoader will be executed. If Connection is not connected, the [Load](#) method calls [TCustomDACConnection.Connect](#).

See Also

- [TCustomDACConnection](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.2.3 TableName Property

Used to specify the name of the table to which data will be loaded.

Class

[TDALoader](#)

Syntax

```
property TableName: string;
```

Remarks

Set the TableName property to specify the name of the table to which data will be loaded. Add TDAColumn objects to [Columns](#) for the fields that are needed to be loaded.

See Also

- [TDAColumn](#)
- [TCustomDACConnection.GetTableNames](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.3 Methods

Methods of the **TDALoader** class.

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

Public

Name	Description
CreateColumns	Creates TDAColumn objects for all fields of the table with the same name as TDALoader.TableName .
Load	Starts loading data.
LoadFromDataSet	Loads data from the specified dataset.
PutColumnData	Overloaded. Puts the value of individual columns.

See Also

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

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.7.1.3.3.1 CreateColumns Method

Creates [TDAColumn](#) objects for all fields of the table with the same name as [TableName](#).

Class

[TDALoader](#)

Syntax

```
procedure CreateColumns;
```

Remarks

Call the CreateColumns method to create [TDAColumn](#) objects for all fields of the table with the same name as [TableName](#). If columns were created before, they will be recreated. You can call CreateColumns from the component popup menu at design-time. After you can customize column loading by setting properties of TDAColumn objects.

See Also

- [TDAColumn](#)
- [TableName](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.3.2 Load Method

Starts loading data.

Class

[TDALoader](#)

Syntax

```
procedure Load; virtual;
```

Remarks

Call the Load method to start loading data. At first it is necessary to [create columns](#) and write one of the [OnPutData](#) or [OnGetColumnData](#) event handlers.

See Also

- [OnGetColumnData](#)
- [OnPutData](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.3.3 LoadFromDataSet Method

Loads data from the specified dataset.

Class

[TDALoader](#)

Syntax

```
procedure LoadFromDataSet(DataSet: TDataSet);
```

Parameters

DataSet

Holds the dataset to load data from.

Remarks

Call the LoadFromDataSet method to load data from the specified dataset. There is no need to create columns and write event handlers for [OnPutData](#) and [OnGetColumnData](#) before calling this method.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.3.4 PutColumnData Method

Puts the value of individual columns.

Class

[TDALoader](#)

Overload List

Name	Description
PutColumnData(Col: integer; Row: integer; const Value: variant)	Puts the value of individual columns by the column index.
PutColumnData(const ColName: string; Row: integer; const Value: variant)	Puts the value of individual columns by the column name.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.4 Events

Events of the **TDALoader** class.

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

Public

Name	Description
OnGetColumnData	Occurs when it is needed to put column values.
OnProgress	Occurs if handling data loading progress of the TDALoader.LoadFromDataSet method is needed.

[OnPutData](#)

Occurs when putting loading data by rows is needed.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.4.1 OnGetColumnData Event

Occurs when it is needed to put column values.

Class

[TDALoader](#)

Syntax

```
property OnGetColumnData: TGetColumnDataEvent;
```

Remarks

Write the OnGetColumnData event handler to put column values. [TDALoader](#) calls the OnGetColumnData event handler for each column in the loop. Column points to a [TDAColumn](#) object that corresponds to the current loading column. Use its Name or Index property to identify what column is loading. The Row parameter indicates the current loading record. TDALoader increments the Row parameter when all the columns of the current record are loaded. The first row is 1. Set EOF to True to stop data loading. Fill the Value parameter by column values. To start loading call the [Load](#) method.

Another way to load data is using the [OnPutData](#) event.

Example

This handler loads 1000 rows.

```
procedure TfmMain.GetColumnData(Sender: TObject;  
    Column: TDAColumn; Row: Integer; var Value: Variant;  
    var EOF: Boolean);  
begin
```

```
if Row <= 1000 then begin
  case Column.Index of
    0: Value := Row;
    1: Value := Random(100);
    2: Value := Random*100;
    3: Value := 'abc01234567890123456789';
    4: Value := Date;
  else
    Value := Null;
  end;
end
else
  EOF := True;
end;
```

See Also

- [OnPutData](#)
- [Load](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.4.2 OnProgress Event

Occurs if handling data loading progress of the [LoadFromDataSet](#) method is needed.

Class

[TDALoader](#)

Syntax

```
property OnProgress: TLoaderProgressEvent;
```

Remarks

Add a handler to this event if you want to handle data loading progress of the [LoadFromDataSet](#) method.

See Also

- [LoadFromDataSet](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.3.4.3 OnPutData Event

Occurs when putting loading data by rows is needed.

Class

[TDALoader](#)

Syntax

```
property OnPutData: TDAPutDataEvent;
```

Remarks

Write the OnPutData event handler to put loading data by rows.

Note that rows should be loaded from the first in the ascending order.

To start loading, call the [Load](#) method. It is more effective way to load data in comparison with using [OnGetColumnData](#). OnPutData event handler must send column data by [TDALoader.PutColumnData](#) method. TDALoader will flush data to Oracle when it is needed.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.4 TDALoaderOptions Class

Allows loading external data into database.

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

Unit

[DALoader](#)

Syntax

```
TDALoaderOptions = class(TPersistent);
```

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.4.1 Members

[TDALoaderOptions](#) class overview.

Properties

Name	Description
UseBlankValues	Forces IBDAC to fill the buffer with null values after loading a row to the database.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.4.2 Properties

Properties of the **TDALoaderOptions** class.

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

Public

Name	Description
UseBlankValues	Forces IBDAC to fill the buffer with null values after loading a row to the database.

See Also

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

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.1.4.2.1 UseBlankValues Property

Forces IBDAC to fill the buffer with null values after loading a row to the database.

Class

[TDALoaderOptions](#)

Syntax

```
property useBlankValues: boolean default True;
```

Remarks

Used to force IBDAC to fill the buffer with null values after loading a row to the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.2 Types

Types in the **DALoader** unit.

Types

Name	Description
TDAPutDataEvent	This type is used for the TDALoader.OnPutData event.
TGetColumnDataEvent	This type is used for the TDALoader.OnGetColumnData event.
TLoaderProgressEvent	This type is used for the TDALoader.OnProgress event.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.2.1 TDAPutDataEvent Procedure Reference

This type is used for the [TDALoader.OnPutData](#) event.

Unit

[DLoader](#)

Syntax

```
TDAPutDataEvent = procedure (Sender: TDALoader) of object;
```

Parameters

Sender

An object that raised the event.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.2.2 TGetColumnDataEvent Procedure Reference

This type is used for the [TDALoader.OnGetColumnData](#) event.

Unit

[DLoader](#)

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.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.7.2.3 TLoaderProgressEvent Procedure Reference

This type is used for the [TDALoader.OnProgress](#) event.

Unit

[DALoader](#)

Syntax

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

Parameters

Sender

An object that raised the event.

Percent

Percentage of the load operation progress.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8 DAScript

This unit contains the base class for the TIBCScript component.

Classes

Name	Description
TDAScript	Makes it possible to execute several SQL statements one by one.
TDAStatement	This class has attributes and methods for controlling single SQL statement of a script.
TDAStatements	Holds a collection of TDAStatement objects.

Types

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

TAfterStatementExecuteEvent	This type is used for the TDAScript.AfterExecute event.
TBeforeStatementExecuteEvent	This type is used for the TDAScript.BeforeExecute event.
TOnErrorEvent	This type is used for the TDAScript.OnError event.

Enumerations

Name	Description
TErrorAction	Indicates the action to take when the OnError handler exits.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1 Classes

Classes in the **DAScript** unit.

Classes

Name	Description
TDAScript	Makes it possible to execute several SQL statements one by one.
TDAStatement	This class has attributes and methods for controlling single SQL statement of a script.
TDAStatements	Holds a collection of TDAStatement objects.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1 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, (for example, CREATE TRIGGER or CREATE PROCEDURE) only slash. Note that slash must be the first character in line.

Errors that occur during execution can be processed in the [TDAScript.OnError](#) event handler. By default, on error TDAScript shows exception and continues execution.

See Also

- [TCustomDASQL](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.1 Members

[TDAScript](#) class overview.

Properties

Name	Description
Connection	Used to specify the connection in which the

	script will be executed.
DataSet	Refers to a dataset that holds the result set of query execution.
Debug	Used to display the script execution and all its parameter values.
Delimiter	Used to set the delimiter string that separates script statements.
EndLine	Used to get the current statement last line number in a script.
EndOffset	Used to get the offset in the last line of the current statement.
EndPos	Used to get the end position of the current statement.
Macros	Used to change SQL script text in design- or run-time easily.
SQL	Used to get or set script text.
StartLine	Used to get the current statement start line number in a script.
StartOffset	Used to get the offset in the first line of the current statement.
StartPos	Used to get the start position of the current statement in a script.
Statements	Contains a list of statements obtained from the SQL property.

Methods

Name	Description
BreakExec	Stops script execution.
ErrorOffset	Used to get the offset of the statement if the Execute method raised an exception.

Execute	Executes a script.
ExecuteFile	Executes SQL statements contained in a file.
ExecuteNext	Executes the next statement in the script and then stops.
ExecuteStream	Executes SQL statements contained in a stream object.
FindMacro	Finds a macro with the specified name.
MacroByName	Finds a macro with the specified name.

Events

Name	Description
AfterExecute	Occurs after a SQL script execution.
BeforeExecute	Occurs when taking a specific action before executing the current SQL statement is needed.
OnError	Occurs when InterBase raises an error.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.2 Properties

Properties of the **TDAScript** class.

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

Public

Name	Description
Connection	Used to specify the connection in which the script will be executed.
DataSet	Refers to a dataset that holds the result set of query

	execution.
EndLine	Used to get the current statement last line number in a script.
EndOffset	Used to get the offset in the last line of the current statement.
EndPos	Used to get the end position of the current statement.
StartLine	Used to get the current statement start line number in a script.
StartOffset	Used to get the offset in the first line of the current statement.
StartPos	Used to get the start position of the current statement in a script.
Statements	Contains a list of statements obtained from the SQL property.

Published

Name	Description
Debug	Used to display the script execution and all its parameter values.
Delimiter	Used to set the delimiter string that separates script statements.
Macros	Used to change SQL script text in design- or run-time easily.
SQL	Used to get or set script text.

See Also

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

Reserved.

5.8.1.1.2.1 Connection Property

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.2.2 DataSet Property

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.2.3 Debug Property

Used to display the script execution and all its parameter values.

Class

[TDAScript](#)

Syntax

```
property Debug: boolean default False;
```

Remarks

Set the Debug property to True to display the statement that is being executed and the values and types of its parameters.

You should add the lbDacVcl unit to the uses clause of any unit in your project to make the Debug property work.

Note: If TIBCSQLMonitor is used in the project and the TIBCSQLMonitor.Active property is set to False, the debug window is not displayed.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.2.4 Delimiter Property

Used to set the delimiter string that separates script statements.

Class

[TDAScript](#)

Syntax

```
property Delimiter: string stored IsDelimiterStored;
```

Remarks

Use the Delimiter property to set the delimiter string that separates script statements. By default it is semicolon (;). You can use slash (/) to separate statements that can contain semicolon (for example, CREATE TRIGGER or CREATE PROCEDURE) if the Delimiter property's default value is semicolon. Note that slash must be the first character in line.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.2.5 EndLine Property

Used to get the current statement last line number in a script.

Class

[TDAScript](#)

Syntax

```
property EndLine: Int64;
```

Remarks

Use the EndLine property to get the current statement last line number in a script.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.2.6 EndOffset Property

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

Class

[TDAScript](#)

Syntax

```
property EndOffset: Int64;
```

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.2.7 EndPos Property

Used to get the end position of the current statement.

Class

[TDAScript](#)

Syntax

```
property EndPos: Int64;
```

Remarks

Use the EndPos property to get the end position of the current statement (the position of the last character in the statement) in a script.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.2.8 Macros Property

Used to change SQL script text in design- or run-time easily.

Class

[TDAScript](#)

Syntax

```
property Macros: TMacros stored False;
```

Remarks

With the help of macros you can easily change SQL script text in design- or run-time. Macros

extend abilities of parameters and allow changing conditions in the WHERE clause or sort order in the ORDER BY clause. You just insert &MacroName in a SQL query text and change value of macro by the Macro property editor in design-time or the MacroByName function in run-time. In time of opening query macro is replaced by its value.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.2.9 SQL Property

Used to get or set script text.

Class

[TDAScript](#)

Syntax

```
property SQL: TStrings;
```

Remarks

Use the SQL property to get or set script text.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.2.10 StartLine Property

Used to get the current statement start line number in a script.

Class

[TDAScript](#)

Syntax

```
property StartLine: Int64;
```

Remarks

Use the StartLine property to get the current statement start line number in a script.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.2.11 StartOffset Property

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

Class

[TDAScript](#)

Syntax

```
property StartOffset: Int64;
```

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.2.12 StartPos Property

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

Class

[TDAScript](#)

Syntax

```
property StartPos: Int64;
```

Remarks

Use the StartPos property to get the start position of the current statement (the position of the first statement character) in a script.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.8.1.1.2.13 Statements Property

Contains a list of statements obtained from the SQL property.

Class

[TDAScript](#)

Syntax

```
property Statements: TDASentences;
```

Remarks

Contains a list of statements that are obtained from the SQL property. Use the Access Statements property to view SQL statement, set parameters or execute the specified statement. Statements is a zero-based array of statement records. Index specifies the array element to access.

For example, consider the following script:

```
CREATE TABLE A (FIELD1 INTEGER);  
INSERT INTO A VALUES(1);  
INSERT INTO A VALUES(2);  
INSERT INTO A VALUES(3);  
CREATE TABLE B (FIELD1 INTEGER);  
INSERT INTO B VALUES(1);  
INSERT INTO B VALUES(2);  
INSERT INTO B VALUES(3);
```

Note: The list of statements is created and filled when the value of Statements property is requested. That's why the first access to the Statements property can take a long time.

Example

You can use the Statements property in the following way:

```
procedure TForm1.Button1Click(Sender: TObject);  
var  
    i: integer;  
begin  
    with Script do  
        begin  
            for i := 0 to Statements.Count - 1 do  
                if Copy(Statements[i].SQL, 1, 6) <> 'CREATE' then  
                    Statements[i].Execute;  
        end;  
end;
```

```
end;
```

See Also

- [TDAStatements](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.3 Methods

Methods of the **TDAScript** class.

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

Public

Name	Description
BreakExec	Stops script execution.
ErrorOffset	Used to get the offset of the statement if the Execute method raised an exception.
Execute	Executes a script.
ExecuteFile	Executes SQL statements contained in a file.
ExecuteNext	Executes the next statement in the script and then stops.
ExecuteStream	Executes SQL statements contained in a stream object.
FindMacro	Finds a macro with the specified name.
MacroByName	Finds a macro with the specified name.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.3.1 BreakExec Method

Stops script execution.

Class

[TDAScript](#)

Syntax

```
procedure BreakExec; virtual;
```

Remarks

Call the BreakExec method to stop script execution.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.3.2 ErrorOffset Method

Used to get the offset of the statement if the Execute method raised an exception.

Class

[TDAScript](#)

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](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.3.3 Execute Method

Executes a script.

Class

[TDAscript](#)

Syntax

```
procedure Execute; virtual;
```

Remarks

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

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.3.4 ExecuteFile Method

Executes SQL statements contained in a file.

Class

[TDAscript](#)

Syntax

```
procedure ExecuteFile(const FileName: string);
```

Parameters

FileName

Holds the file name.

Remarks

Call the ExecuteFile method to execute SQL statements contained in a file. Script doesn't load full content into memory. Reading and execution is performed by blocks of 64k size. Therefore, it is optimal to use it for big files.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.3.5 ExecuteNext Method

Executes the next statement in the script and then stops.

Class

[TDAScript](#)

Syntax

```
function ExecuteNext: boolean; virtual;
```

Return Value

True, if there are any statements left in the script, False otherwise.

Remarks

Use the ExecuteNext method to execute the next statement in the script statement and stop. If InterBase raises an error, the OnError event occurs.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.3.6 ExecuteStream Method

Executes SQL statements contained in a stream object.

Class

[TDAScript](#)

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.3.7 FindMacro Method

Finds a macro with the specified name.

Class

[TDAScript](#)

Syntax

```
function FindMacro(Name: string): TMacro;
```

Parameters

Name

Holds the name of a macro to search for.

Return Value

TMacro object if a match is found, nil otherwise.

Remarks

Call the FindMacro method to find a macro with the specified name. If a match is found, FindMacro returns the macro. Otherwise, it returns nil. Use this method instead of a direct reference to the [TMacros.Items](#) property to avoid depending on the order of the items.

See Also

- [TMacro](#)

- [Macros](#)
- [MacroByName](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.3.8 MacroByName Method

Finds a macro with the specified name.

Class

[TDAScript](#)

Syntax

```
function MacroByName(Name: string): TMacro;
```

Parameters

Name

Holds the name of a macro to search for.

Return Value

TMacro object if a match is found.

Remarks

Call the MacroByName method to find a macro with the specified name. If a match is found, MacroByName returns the macro. Otherwise, an exception is raised. Use this method instead of a direct reference to the [TMacros.Items](#) property to avoid depending on the order of the items.

To locate a parameter by name without raising an exception if the parameter is not found, use the FindMacro method.

To set a value to a macro, use the [TMacro.Value](#) property.

See Also

- [TMacro](#)
- [Macros](#)
- [FindMacro](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.4 Events

Events of the **TDAScript** class.

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

Published

Name	Description
AfterExecute	Occurs after a SQL script execution.
BeforeExecute	Occurs when taking a specific action before executing the current SQL statement is needed.
OnError	Occurs when InterBase raises an error.

See Also

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

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.4.1 AfterExecute Event

Occurs after a SQL script execution.

Class

[TDAScript](#)

Syntax

```
property AfterExecute: TAfterStatementExecuteEvent;
```

Remarks

Occurs after a SQL script has been executed.

See Also

- [Execute](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.4.2 BeforeExecute Event

Occurs when taking a specific action before executing the current SQL statement is needed.

Class

[TDAScript](#)

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.1.4.3 OnError Event

Occurs when InterBase raises an error.

Class

[TDAScript](#)

Syntax

```
property OnError: TOnErrorEvent;
```

Remarks

Occurs when InterBase raises an error.

Action indicates the action to take when the OnError handler exits. On entry into the handler, Action is always set to eaFail.

See Also

- [ErrorOffset](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2 TDASStatement 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 [TDASStatement](#) 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.

See Also

- [TDAScript](#)
- [TDASStatements](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.1 Members

[TDASStatement](#) class overview.

Properties

Name	Description
EndLine	Used to determine the number of the last statement line in a script.
EndOffset	Used to get the offset in the last line of the statement.
EndPos	Used to get the end position of the statement in a script.
Omit	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.

Methods

Name	Description
Execute	Executes a statement.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.2 Properties

Properties of the **TDASstatement** class.

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

Public

Name	Description
EndLine	Used to determine the number of the last statement line in a script.
EndOffset	Used to get the offset in the last line of the statement.
EndPos	Used to get the end position of the statement in a script.
Omit	Used to avoid execution of a statement.
Params	Contains parameters for an SQL statement.
Script	Used to determine the TDA Script object the SQL Statement belongs to.
SQL	Used to get or set the text of an SQL statement.
StartLine	Used to determine the number of the first statement line in a script.
StartOffset	Used to get the offset in the first line of a statement.
StartPos	Used to get the start position of the statement in a script.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.2.1 EndLine Property

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

Class

[TDASTatement](#)

Syntax

```
property EndLine: integer;
```

Remarks

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

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.2.2 EndOffset Property

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.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.2.3 EndPos Property

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.2.4 Omit Property

Used to avoid execution of a statement.

Class

[TDASstatement](#)

Syntax

```
property Omit: boolean;
```

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.2.5 Params Property

Contains parameters for an SQL statement.

Class

[TDASstatement](#)

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.2.6 Script Property

Used to determine the TDAScript object the SQL Statement belongs to.

Class

[TDASstatement](#)

Syntax

```
property script: TDAScript;
```

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.2.7 SQL Property

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

Class

[TDASstatement](#)

Syntax

```
property SQL: string;
```

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.2.8 StartLine Property

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

Class

[TDASstatement](#)

Syntax

```
property startLine: integer;
```

Remarks

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

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.2.9 StartOffset Property

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

Class

[TDASstatement](#)

Syntax

```
property startOffset: integer;
```

Remarks

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

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.2.10 StartPos Property

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

Class

[TDASstatement](#)

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.3 Methods

Methods of the **TDASStatement** class.

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

Public

Name	Description
Execute	Executes a statement.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.2.3.1 Execute Method

Executes a statement.

Class

[TDASStatement](#)

Syntax

```
procedure Execute;
```

Remarks

Use the Execute method to execute a statement.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.3 TDAStatements Class

Holds a collection of [TDASStatement](#) objects.

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

Unit

[DAScript](#)

Syntax

```
TDAStatements = class(TCollection);
```

Remarks

Each TDAStatements holds a collection of [TDASStatement](#) objects. TDAStatements maintains an index of the statements in its Items array. The Count property contains the number of statements in the collection. Use TDAStatements class to manipulate script SQL statements.

See Also

- [TDAScript](#)
- [TDASStatement](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.3.1 Members

[TDAStatements](#) class overview.

Properties

Name	Description
Items	Used to access separate script statements.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.3.2 Properties

Properties of the **TDAStatements** class.

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

Public

Name	Description
Items	Used to access separate script statements.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.1.3.2.1 Items Property(Indexer)

Used to access separate script statements.

Class

[TDAStatements](#)

Syntax

```
property Items[Index: Integer]: TDASatement; default t;
```

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 [TDASStatement](#).

See Also

- [TDASStatement](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.2 Types

Types in the **DAScript** unit.

Types

Name	Description
TAfterStatementExecuteEvent	This type is used for the TDAScript.AfterExecute event.
TBeforeStatementExecuteEvent	This type is used for the TDAScript.BeforeExecute event.
TOnErrorEvent	This type is used for the TDAScript.OnError event.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.2.1 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.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.2.2 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.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.2.3 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.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.3 Enumerations

Enumerations in the **DAScript** unit.

Enumerations

Name	Description
TErrorAction	Indicates the action to take when the OnError handler exits.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.8.3.1 TErrorAction Enumeration

Indicates the action to take when the OnError handler exits.

Unit

[DAScript](#)

Syntax

```
TErrorAction = (eaAbort, eaFail, eaException, eaContinue);
```

Values

Value	Meaning
eaAbort	Abort execution without displaying an error message.
eaContinue	Continue execution.
eaException	In Delphi 6 and higher exception is handled by the Application.HandleException method.
eaFail	Abort execution and display an error message.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.9 DASQLMonitor

This unit contains the base class for the TIBCSQLMonitor component.

Classes

Name	Description
TCustomDASQLMonitor	A base class that introduces properties and methods to monitor dynamic SQL execution in database applications interactively.
TDBMonitorOptions	This class holds options for dbMonitor.

Types

Name	Description
TDATraceFlags	Represents the set of TDATraceFlag .
TMonitorOptions	Represents the set of TMonitorOption .
TOnSQLEvent	This type is used for the TCustomDASQLMonitor.OnSQL event.

Enumerations

Name	Description
TDATraceFlag	Use TraceFlags to specify which database operations

	the monitor should track in an application at runtime.
TMonitorOption	Used to define where information from SQLMonitor will be displayed.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1 Classes

Classes in the **DASQLMonitor** unit.

Classes

Name	Description
TCustomDASQLMonitor	A base class that introduces properties and methods to monitor dynamic SQL execution in database applications interactively.
TDBMonitorOptions	This class holds options for dbMonitor.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.1 TCustomDASQLMonitor Class

A base class that introduces properties and methods to monitor dynamic SQL execution in database applications interactively.

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

Unit

[DASQLMonitor](#)

Syntax

```
TCustomDASQLMonitor = class(TComponent);
```

Remarks

TCustomDASQLMonitor is a base class that introduces properties and methods to monitor dynamic SQL execution in database applications interactively. TCustomDASQLMonitor provides two ways of displaying debug information. It monitors either by dialog window or by Borland's proprietary SQL Monitor. Furthermore to receive debug information use the [TCustomDASQLMonitor.OnSQL](#) event.

In applications use descendants of TCustomDASQLMonitor.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.1.1 Members

[TCustomDASQLMonitor](#) class overview.

Properties

Name	Description
Active	Used to activate monitoring of SQL.
DBMonitorOptions	Used to set options for dbMonitor.
Options	Used to include the desired properties for TCustomDASQLMonitor.
TraceFlags	Used to specify which database operations the monitor should track in an application at runtime.

Events

Name	Description
OnSQL	Occurs when tracing of SQL activity on database components is needed.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.1.2 Properties

Properties of the **TCustomDASQLMonitor** class.

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

Public

Name	Description
Active	Used to activate monitoring of SQL.
DBMonitorOptions	Used to set options for dbMonitor.
Options	Used to include the desired properties for TCustomDASQLMonitor.
TraceFlags	Used to specify which database operations the monitor should track in an application at runtime.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.1.2.1 Active Property

Used to activate monitoring of SQL.

Class

[TCustomDASQLMonitor](#)

Syntax

```
property Active: boolean default True;
```

Remarks

Set the Active property to True to activate monitoring of SQL.

See Also

- [OnSQL](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.1.2.2 DBMonitorOptions Property

Used to set options for dbMonitor.

Class

[TCustomDASQLMonitor](#)

Syntax

```
property DBMonitorOptions: TDBMonitorOptions;
```

Remarks

Use DBMonitorOptions to set options for dbMonitor.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.1.2.3 Options Property

Used to include the desired properties for TCustomDASQLMonitor.

Class

[TCustomDASQLMonitor](#)

Syntax

```
property Options: TMonitorOptions default [moDialog,  
moSQLMonitor, moDBMonitor, moCustom];
```

Remarks

Set Options to include the desired properties for TCustomDASQLMonitor.

See Also

- [OnSQL](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.1.2.4 TraceFlags Property

Used to specify which database operations the monitor should track in an application at runtime.

Class

[TCustomDASQLMonitor](#)

Syntax

```
property TraceFlags: TDATraceFlags default [tfQPrepare, tfQExecute, tfError, tfConnect, tfTransact, tfParams, tfMisc];
```

Remarks

Use the TraceFlags property to specify which database operations the monitor should track in an application at runtime.

See Also

- [OnSQL](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.1.3 Events

Events of the **TCustomDASQLMonitor** class.

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

Public

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

[OnSQL](#)

Occurs when tracing of SQL activity on database components is needed.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.1.3.1 OnSQL Event

Occurs when tracing of SQL activity on database components is needed.

Class

[TCustomDASQLMonitor](#)

Syntax

```
property OnSQL: TOnSQLEvent;
```

Remarks

Write the OnSQL event handler to let an application trace SQL activity on database components. The Text parameter holds the detected SQL statement. Use the Flag parameter to make selective processing of SQL in the handler body.

See Also

- [TraceFlags](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.2 TDBMonitorOptions Class

This class holds options for dbMonitor.

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

Unit

[DASQLMonitor](#)

Syntax

```
TDBMonitorOptions = class(TPersistent);
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.2.1 Members

[TDBMonitorOptions](#) class overview.

Properties

Name	Description
Host	Used to set the host name or IP address of the computer where dbMonitor application runs.
Port	Used to set the port number for connecting to dbMonitor.
ReconnectTimeout	Used to set the minimum time that should be spent before reconnecting to dbMonitor is allowed.
SendTimeout	Used to set timeout for sending events to dbMonitor.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.2.2 Properties

Properties of the **TDBMonitorOptions** class.

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

Published

Name	Description
Host	Used to set the host name or IP address of the computer where dbMonitor application runs.
Port	Used to set the port number for connecting to dbMonitor.
ReconnectTimeout	Used to set the minimum time that should be spent before reconnecting to dbMonitor is allowed.
SendTimeout	Used to set timeout for sending events to dbMonitor.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.2.2.1 Host Property

Used to set the host name or IP address of the computer where dbMonitor application runs.

Class

[TDBMonitorOptions](#)

Syntax

```
property Host: string;
```

Remarks

Use the Host property to set the host name or IP address of the computer where dbMonitor application runs.

dbMonitor supports remote monitoring. You can run dbMonitor on a different computer than monitored application runs. In this case you need to set the Host property to the corresponding computer name.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.2.2.2 Port Property

Used to set the port number for connecting to dbMonitor.

Class

[TDBMonitorOptions](#)

Syntax

```
property Port: integer default DBMonitorPort;
```

Remarks

Use the Port property to set the port number for connecting to dbMonitor.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.1.2.2.3 ReconnectTimeout Property

Used to set the minimum time that should be spent before reconnecting to dbMonitor is allowed.

Class

[TDBMonitorOptions](#)

Syntax

```
property ReconnectTimeout: integer default  
DefaultReconnectTimeout;
```

Remarks

Use the ReconnectTimeout property to set the minimum time (in milliseconds) that should be spent before allowing reconnecting to dbMonitor. If an error occurs when the component sends an event to dbMonitor (dbMonitor is not running), next events are ignored and the component does not restore the connection until ReconnectTimeout is over.

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.9.1.2.2.4 SendTimeout Property

Used to set timeout for sending events to dbMonitor.

Class

[TDBMonitorOptions](#)

Syntax

```
property SendTimeout: integer default DefaultSendTimeout;
```

Remarks

Use the SendTimeout property to set timeout (in milliseconds) for sending events to dbMonitor. If dbMonitor does not respond in the specified timeout, event is ignored.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.2 Types

Types in the **DASQLMonitor** unit.

Types

Name	Description
TDATraceFlags	Represents the set of TDATraceFlag .
TMonitorOptions	Represents the set of TMonitorOption .
TOnSQLEvent	This type is used for the TCustomDASQLMonitor.OnSQL event.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.2.1 TDATraceFlags Set

Represents the set of [TDATraceFlag](#).

Unit

[DASQLMonitor](#)

Syntax

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

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.2.2 TMonitorOptions Set

Represents the set of [TMonitorOption](#).

Unit

[DASQLMonitor](#)

Syntax

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

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.2.3 TOnSQLEvent Procedure Reference

This type is used for the [TCustomDASQLMonitor.OnSQL](#) event.

Unit

[DASQLMonitor](#)

Syntax

```
TOnSQLEvent = procedure (Sender: TObject; Text: string; Flag:  
TDATraceFlag) of object;
```

Parameters

Sender

An object that raised the event.

Text

Holds the detected SQL statement.

Flag

Use the Flag parameter to make selective processing of SQL in the handler body.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.3 Enumerations

Enumerations in the **DASQLMonitor** unit.

Enumerations

Name	Description
TDATraceFlag	Use TraceFlags to specify which database operations the monitor should track in an application at runtime.
TMonitorOption	Used to define where information from SQLMonitor will be dispalyed.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.3.1 TDATraceFlag Enumeration

Use TraceFlags to specify which database operations the monitor should track in an application at runtime.

Unit

[DASQLMonitor](#)

Syntax

```
TDATraceFlag = (tfQPrepare, tfQExecute, tfQFetch, tfError, tfStmt,
tfConnect, tfTransact, tfBlob, tfService, tfMisc, tfParams,
tfObjDestroy, tfPool);
```

Values

Value	Meaning
tfBlob	This option is declared for future use.
tfConnect	Establishing a connection.
tfError	Errors of query execution.
tfMisc	This option is declared for future use.
tfObjDestroy	Destroying of components.
tfParams	Representing parameter values for tfQPrepare and tfQExecute.
tfPool	Connection pool operations.
tfQExecute	Execution of the queries.
tfQFetch	This option is declared for future use.
tfQPrepare	Queries preparation.
tfService	This option is declared for future use.
tfStmt	This option is declared for future use.
tfTransact	Processing transactions.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.9.3.2 TMonitorOption Enumeration

Used to define where information from SQLMonitor will be displayed.

Unit

[DASQLMonitor](#)

Syntax

```
TMonitorOption = (moDialog, moSQLMonitor, moDBMonitor, moCustom, moHandled);
```

Values

Value	Meaning
moCustom	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.
moDBMonitor	Debug information is displayed in DBMonitor .

moDialog	Debug information is displayed in debug window.
moHandled	Component handle is included into the event description string.
moSQLMonitor	Debug information is displayed in Borland SQL Monitor.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10 DBAccess

This unit contains base classes for most of the components.

Classes

Name	Description
EDAEError	A base class for exceptions that are raised when an error occurs on the server side.
TCRDataSource	Provides an interface between a DAC dataset components and data-aware controls on a form.
TCustomConnectDialog	A base class for the connect dialog components.
TCustomDAConnection	A base class for components used to establish connections.
TCustomDADataset	Encapsulates general set of properties, events, and methods for working with data accessed through various database engines.
TCustomDASQL	A base class for components executing SQL statements that do not return result sets.
TCustomDAUpdateSQL	A base class for components that provide DML statements for more flexible control over data modifications.
TDACCondition	Represents a condition from the TDACConditions list.

TDAConditions	Holds a collection of TDACondition objects.
TDAConnectionOptions	This class allows setting up the behaviour of the TDAConnection class.
TDAConnectionSSLOptions	This class is used to set up the SSL options.
TDADatasetOptions	This class allows setting up the behaviour of the TDADataset class.
TDAEncryption	Used to specify the options of the data encryption in a dataset.
TDAMapRule	Class that forms rules for Data Type Mapping.
TDAMapRules	Used for adding rules for DataSet fields mapping with both identifying by field name and by field type and Delphi field types.
TDAMetaData	A class for retrieving metainformation of the specified database objects in the form of dataset.
TDAParam	A class that forms objects to represent the values of the parameters set .
TDAParams	This class is used to manage a list of TDAParam objects for an object that uses field parameters.
TDATransaction	A base class that implements functionality for controlling transactions.
TMacro	Object that represents the value of a macro.
TMacros	Controls a list of TMacro objects for the TCustomDASQL.Macros or TCustomDADataset components.
TPoolingOptions	This class allows setting up the behaviour of the connection pool.

TSmartFetchOptions	Smart fetch options are used to set up the behavior of the SmartFetch mode.
------------------------------------	---

Types

Name	Description
TAfterExecuteEvent	This type is used for the TCustomDADataset.AfterExecute and TCustomDASQL.AfterExecute events.
TAfterFetchEvent	This type is used for the TCustomDADataset.AfterFetch event.
TBeforeFetchEvent	This type is used for the TCustomDADataset.BeforeFetch event.
TConnectionLostEvent	This type is used for the TCustomDAConnection.OnConnectionLost event.
TDAConnectionErrorEvent	This type is used for the TCustomDAConnection.OnError event.
TDATransactionErrorEvent	This type is used for the TDATransaction.OnError event.
TRefreshOptions	Represents the set of TRefreshOption .
TUpdateExecuteEvent	This type is used for the TCustomDADataset.AfterUpdateExecute and TCustomDADataset.BeforeUpdateExecute events.

Enumerations

Name	Description
TLabelSet	Sets the language of labels in the connect dialog.
TLockMode	Specifies the lock mode.
TRefreshOption	Indicates when the editing

	record will be refreshed.
TRetryMode	Specifies the application behavior when connection is lost.

Variables

Name	Description
ChangeCursor	When set to True allows data access components to change screen cursor for the execution time.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1 Classes

Classes in the **DBAccess** unit.

Classes

Name	Description
EDAError	A base class for exceptions that are raised when an error occurs on the server side.
TCRDataSource	Provides an interface between a DAC dataset components and data-aware controls on a form.
TCustomConnectDialog	A base class for the connect dialog components.
TCustomDACConnection	A base class for components used to establish connections.
TCustomDADataset	Encapsulates general set of properties, events, and methods for working with data accessed through various database engines.
TCustomDASQL	A base class for components executing SQL statements that do not return

	result sets.
TCustomDAUpdateSQL	A base class for components that provide DML statements for more flexible control over data modifications.
TDACondition	Represents a condition from the TDAConditions list.
TDAConditions	Holds a collection of TDACondition objects.
TDAConnectionOptions	This class allows setting up the behaviour of the TDAConnection class.
TDAConnectionSSLOptions	This class is used to set up the SSL options.
TDADatasetOptions	This class allows setting up the behaviour of the TDADataset class.
TDAEncryption	Used to specify the options of the data encryption in a dataset.
TDAMapRule	Class that forms rules for Data Type Mapping.
TDAMapRules	Used for adding rules for DataSet fields mapping with both identifying by field name and by field type and Delphi field types.
TDAMetaData	A class for retrieving metainformation of the specified database objects in the form of dataset.
TDAParam	A class that forms objects to represent the values of the parameters set .
TDAParams	This class is used to manage a list of TDAParam objects for an object that uses field parameters.
TDATransaction	A base class that implements functionality for controlling transactions.
TMacro	Object that represents the value of a macro.

TMacros	Controls a list of TMacro objects for the TCustomDASQL.Macros or TCustomDADataset components.
TPoolingOptions	This class allows setting up the behaviour of the connection pool.
TSmartFetchOptions	Smart fetch options are used to set up the behavior of the SmartFetch mode.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.1 EDAError Class

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

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

Unit

[DBAccess](#)

Syntax

```
EDAError = class(EDatabaseError);
```

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.1.1 Members

[EDAError](#) class overview.

Properties

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

Component	Contains the component that caused the error.
ErrorCode	Determines the error code returned by the server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.1.2 Properties

Properties of the **EDAEError** class.

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

Public

Name	Description
Component	Contains the component that caused the error.
ErrorCode	Determines the error code returned by the server.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.1.2.1 Component Property

Contains the component that caused the error.

Class

[EDAEError](#)

Syntax

```
property Component: TObject;
```

Remarks

The Component property contains the component that caused the error.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.1.2.2 ErrorCode Property

Determines the error code returned by the server.

Class

[EDAError](#)

Syntax

```
property ErrorCode: integer;
```

Remarks

Use the ErrorCode property to determine the error code returned by InterBase. This value is always positive.

See Also

- [EIBCError.ErrorNumber](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.2 TCRDataSource Class

Provides an interface between a DAC dataset components and data-aware controls on a form.

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

Unit

[DBAccess](#)

Syntax

```
TCRDataSource = class(TDataSource);
```

Remarks

TCRDataSource provides an interface between a DAC dataset components and data-aware controls on a form.

TCRDataSource inherits its functionality directly from the TDataSource component.

At design time assign individual data-aware components' DataSource properties from their drop-down listboxes.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.2.1 Members

[TCRDataSource](#) class overview.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3 TCustomConnectDialog Class

A base class for the connect dialog components.

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

Unit

[DBAccess](#)

Syntax

```
TCustomConnectDialog = class(TComponent);
```

Remarks

TCustomConnectDialog is a base class for the connect dialog components. It provides functionality to show a dialog box where user can edit username, password and server name before connecting to a database. You can customize captions of buttons and labels by their properties.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.1 Members

[TCustomConnectDialog](#) class overview.

Properties

Name	Description
CancelButton	Used to specify the label for the Cancel button.
Caption	Used to set the caption of dialog box.
ConnectButton	Used to specify the label for the Connect button.
DialogClass	Used to specify the class of the form that will be displayed to enter login information.
LabelSet	Used to set the language of buttons and labels captions.
PasswordLabel	Used to specify a prompt for password edit.
Retries	Used to indicate the number of retries of failed connections.
SavePassword	Used for the password to be displayed in ConnectDialog in asterisks.
ServerLabel	Used to specify a prompt for the server name edit.
StoreLogInfo	Used to specify whether the login information should be kept in system registry after a connection was established.
UsernameLabel	Used to specify a prompt for username edit.

Methods

Name	Description
Execute	Displays the connect dialog and calls the connection's Connect method when user

	clicks the Connect button.
GetServerList	Retrieves a list of available server names.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.2 Properties

Properties of the **TCustomConnectDialog** class.

For a complete list of the **TCustomConnectDialog** class members, see the

[TCustomConnectDialog Members](#) topic.

Public

Name	Description
CancelButton	Used to specify the label for the Cancel button.
Caption	Used to set the caption of dialog box.
ConnectButton	Used to specify the label for the Connect button.
DialogClass	Used to specify the class of the form that will be displayed to enter login information.
LabelSet	Used to set the language of buttons and labels captions.
PasswordLabel	Used to specify a prompt for password edit.
Retries	Used to indicate the number of retries of failed connections.
SavePassword	Used for the password to be displayed in ConnectDialog in asterisks.
ServerLabel	Used to specify a prompt for the server name edit.
StoreLogInfo	Used to specify whether the login information should be kept in system registry after a connection was

	established.
UsernameLabel	Used to specify a prompt for username edit.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.2.1 CancelButton Property

Used to specify the label for the Cancel button.

Class

[TCustomConnectDialog](#)

Syntax

```
property cancelButton: string;
```

Remarks

Use the CancelButton property to specify the label for the Cancel button.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.2.2 Caption Property

Used to set the caption of dialog box.

Class

[TCustomConnectDialog](#)

Syntax

```
property caption: string;
```

Remarks

Use the Caption property to set the caption of dialog box.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.2.3 ConnectButton Property

Used to specify the label for the Connect button.

Class

[TCustomConnectDialog](#)

Syntax

```
property ConnectButton: string;
```

Remarks

Use the ConnectButton property to specify the label for the Connect button.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.2.4 DialogClass Property

Used to specify the class of the form that will be displayed to enter login information.

Class

[TCustomConnectDialog](#)

Syntax

```
property DialogClass: string;
```

Remarks

Use the DialogClass property to specify the class of the form that will be displayed to enter login information. When this property is blank, TCustomConnectDialog uses the default form - TConnectForm. You can write your own login form to enter login information and assign its class name to the DialogClass property. Each login form must have 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 IBDAC installation directory.

See Also

- [GetServerList](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.2.5 LabelSet Property

Used to set the language of buttons and labels captions.

Class

[TCustomConnectDialog](#)

Syntax

```
property LabelSet: TLabelSet default IsEnglish;
```

Remarks

Use the LabelSet property to set the language of labels and buttons captions.

The default value is IsEnglish.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.2.6 PasswordLabel Property

Used to specify a prompt for password edit.

Class

[TCustomConnectDialog](#)

Syntax

```
property PasswordLabel: string;
```

Remarks

Use the PasswordLabel property to specify a prompt for password edit.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.2.7 Retries Property

Used to indicate the number of retries of failed connections.

Class

[TCustomConnectDialog](#)

Syntax

```
property Retries: word default 3;
```

Remarks

Use the Retries property to determine the number of retries of failed connections.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.2.8 SavePassword Property

Used for the password to be displayed in ConnectDialog in asterisks.

Class

[TCustomConnectDialog](#)

Syntax

```
property SavePassword: boolean default False;
```

Remarks

If True, and the Password property of the connection instance is assigned, the password in ConnectDialog is displayed in asterisks.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.10.1.3.2.9 ServerLabel Property

Used to specify a prompt for the server name edit.

Class

[TCustomConnectDialog](#)

Syntax

```
property ServerLabel: string;
```

Remarks

Use the ServerLabel property to specify a prompt for the server name edit.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.2.10 StoreLogInfo Property

Used to specify whether the login information should be kept in system registry after a connection was established.

Class

[TCustomConnectDialog](#)

Syntax

```
property StoreLogInfo: boolean default True;
```

Remarks

Use the StoreLogInfo property to specify whether to keep login information in system registry after a connection was established using provided username, password and servername.

Set this property to True to store login information.

The default value is True.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.2.11 UsernameLabel Property

Used to specify a prompt for username edit.

Class

[TCustomConnectDialog](#)

Syntax

```
property UsernameLabel: string;
```

Remarks

Use the UsernameLabel property to specify a prompt for username edit.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.3 Methods

Methods of the **TCustomConnectDialog** class.

For a complete list of the **TCustomConnectDialog** class members, see the

[TCustomConnectDialog Members](#) topic.

Public

Name	Description
Execute	Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button.
GetServerList	Retrieves a list of available server names.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.3.1 Execute Method

Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button.

Class

[TCustomConnectDialog](#)

Syntax

```
function Execute: boolean; virtual;
```

Return Value

True, if connected.

Remarks

Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button. Returns True if connected. If user clicks Cancel, Execute returns False.

In the case of failed connection Execute offers to connect repeat [Retries](#) times.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.3.3.2 GetServerList Method

Retrieves a list of available server names.

Class

[TCustomConnectDialog](#)

Syntax

```
procedure GetServerList(List: TStrings); virtual;
```

Parameters

List

Holds a list of available server names.

Remarks

Call the GetServerList method to retrieve a list of available server names. It is particularly

relevant for writing custom login form.

See Also

- [DialogClass](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4 TCustomDACConnection Class

A base class for components used to establish connections.

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

Unit

[DBAccess](#)

Syntax

```
TCustomDACConnection = class(TCustomConnection);
```

Remarks

TCustomDACConnection is a base class for components that establish connection with database, provide customised login support, and perform transaction control.

Do not create instances of TCustomDACConnection. To add a component that represents a connection to a source of data, use descendants of the TCustomDACConnection class.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.1 Members

[TCustomDACConnection](#) class overview.

Properties

Name	Description
ConnectDialog	Allows to link a TCustomConnectDialog component.

ConnectionString	Used to specify the connection information, such as: UserName, Password, Server, etc.
ConvertEOL	Allows customizing line breaks in string fields and parameters.
InTransaction	Indicates whether the transaction is active.
LoginPrompt	Specifies whether a login dialog appears immediately before opening a new connection.
Options	Specifies the connection behavior.
Password	Serves to supply a password for login.
Pooling	Enables or disables using connection pool.
PoolingOptions	Specifies the behaviour of connection pool.
Server	Serves to supply the server name for login.
Username	Used to supply a user name for login.

Methods

Name	Description
ApplyUpdates	Overloaded. Applies changes in datasets.
Commit	Commits current transaction.
Connect	Establishes a connection to the server.
CreateSQL	Creates a component for queries execution.
Disconnect	Performs disconnect.
ExecProc	Allows to execute stored procedure or function providing its name and parameters.
ExecProcEx	Allows to execute a stored

	procedure or function.
ExecSQL	Executes a SQL statement with parameters.
ExecSQLEx	Executes any SQL statement outside the TQuery or TSQL components.
GetDatabaseNames	Returns a database list from the server.
GetKeyFieldNames	Provides a list of available key field names.
GetStoredProcNames	Returns a list of stored procedures from the server.
GetTableNames	Provides a list of available tables names.
MonitorMessage	Sends a specified message through the TCustomDASQLMonitor component.
Ping	Used to check state of connection to the server.
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.

Events

Name	Description
OnConnectionLost	This event occurs when connection was lost.
OnError	This event occurs when an error has arisen in the connection.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.2 Properties

Properties of the **TCustomDACConnection** class.

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

Public

Name	Description
ConnectDialog	Allows to link a TCustomConnectDialog component.
ConnectionString	Used to specify the connection information, such as: UserName, Password, Server, etc.
ConvertEOL	Allows customizing line breaks in string fields and parameters.
InTransaction	Indicates whether the transaction is active.
LoginPrompt	Specifies whether a login dialog appears immediately before opening a new connection.
Options	Specifies the connection behavior.
Password	Serves to supply a password for login.
Pooling	Enables or disables using connection pool.
PoolingOptions	Specifies the behaviour of connection pool.
Server	Serves to supply the server name for login.
Username	Used to supply a user name for login.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.2.1 ConnectDialog Property

Allows to link a [TCustomConnectDialog](#) component.

Class

[TCustomDAConnection](#)

Syntax

```
property ConnectDialog: TCustomConnectDialog;
```

Remarks

Use the ConnectDialog property to assign to connection a [TCustomConnectDialog](#) component.

See Also

- [TCustomConnectDialog](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.2.2 ConnectString Property

Used to specify the connection information, such as: UserName, Password, Server, etc.

Class

[TCustomDAConnection](#)

Syntax

```
property ConnectString: string stored False;
```

Remarks

IBDAC recognizes an ODBC-like syntax in provider string property values. Within the string, elements are delimited by using a semicolon. Each element consists of a keyword, an equal sign character, and the value passed on initialization. For example:

```
Server=London1;User ID=nancyd
```

Connection parameters

The following connection parameters can be used to customize connection:

Parameter Name	Description
LoginPrompt	Specifies whether a login dialog appears immediately before opening a new connection.
Pooling	Enables or disables using connection pool.
ConnectionLifeTime	Used to specify the maximum time during which an opened connection can be used by connection pool.
MaxPoolSize	Used to specify the maximum number of connections that can be opened in connection pool.
MinPoolSize	Used to specify the minimum number of connections that can be opened in connection pool.
Validate Connection	Used for a connection to be validated when it is returned from the pool.
Server	Serves to supply the server name for login.
Username	Used to supply a user name for login.
Password	Used to supply a user name for login.
ClientLibrary	Used to set or get the client library location.
Database	Used to set the name of the database to associate with TIBCConnection component.
Charset	Used to set the character set that IBDAC uses to read and write character data.
Protocol	Used to specify the Network protocol of connection with InterBase server.
Role	Used to specify the InterBase connection role.
UseUnicode	Used to enable or disable Unicode support.
Port	Used to specify the port number for the connection.

See Also

- [Password](#)

- [Username](#)
- [Server](#)
- [Connect](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.2.3 ConvertEOL Property

Allows customizing line breaks in string fields and parameters.

Class

[TCustomDACConnection](#)

Syntax

```
property ConvertEOL: boolean default False;
```

Remarks

Affects the line break behavior in string fields and parameters. When fetching strings (including the text BLOB fields) with ConvertEOL = True, dataset converts their line breaks from the LF to CRLF form. And when posting strings to server with ConvertEOL turned on, their line breaks are converted from CRLF to LF form. By default, strings are not converted.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.2.4 InTransaction Property

Indicates whether the transaction is active.

Class

[TCustomDACConnection](#)

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.2.5 LoginPrompt Property

Specifies whether a login dialog appears immediately before opening a new connection.

Class

[TCustomDACConnection](#)

Syntax

```
property LoginPrompt default DefValLoginPrompt;
```

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 lbDacVcl unit appears to the uses clause.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.2.6 Options Property

Specifies the connection behavior.

Class

[TCustomDACConnection](#)

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
AllowImplicitConnect	Specifies whether to allow or not implicit connection opening.
DefaultSortType	Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the TMemDataSet.IndexFieldNames property of a dataset.
DisconnectedMode	Used to open a connection only when needed for performing a server call and closes after performing the operation.
KeepDesignConnected	Used to prevent an application from establishing a connection at the time of startup.
LocalFailover	If True, the OnConnectionLost event occurs and a failover operation can be performed after connection breaks.

See Also

- [Disconnected Mode](#)
- [Working in an Unstable Network](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.2.7 Password Property

Serves to supply a password for login.

Class

[TCustomDAConnection](#)

Syntax

```
property Password: string stored False;
```

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.2.8 Pooling Property

Enables or disables using connection pool.

Class

[TCustomDAConnection](#)

Syntax

```
property Pooling: boolean default DefValPooling;
```

Remarks

Normally, when TCustomDAConnection establishes connection with the server it takes server memory and time resources for allocating new server connection. For example, pooling can be very useful when using disconnect mode. If an application has wide user activity that forces many connect/disconnect operations, it may spend a lot of time on creating connection and sending requests to the server. TCustomDAConnection has software pool which stores open connections with identical parameters.

Connection pool uses separate thread that validates the pool every 30 seconds. Pool

validation consists of checking each connection in the pool. If a connection is broken due to a network problem or another reason, it is deleted from the pool. The validation procedure removes also connections that are not used for a long time even if they are valid from the pool.

Set Pooling to True to enable pooling. Specify correct values for PoolingOptions. Two connections belong to the same pool if they have identical values for the parameters: [MinPoolSize](#), [MaxPoolSize](#), [Validate](#), [ConnectionLifeTime](#), [TIBCCConnection.Database](#), [TIBCCConnectionOptions.Charset](#), [TIBCCConnectionOptions.UseUnicode](#), [TIBCCConnectionOptions.Role](#), [TIBCCConnection.SQLDialect](#), [TIBCCConnection.Params](#).

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

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.2.9 PoolingOptions Property

Specifies the behaviour of connection pool.

Class

[TCustomDAConnection](#)

Syntax

```
property PoolingOptions: TPoolingOptions;
```

Remarks

Set the properties of PoolingOptions to specify the behaviour of connection pool.

Descriptions of all options are in the table below.

Option Name	Description
ConnectionLifetime	Used to specify the maximum time during which an open connection can be used by connection pool.
MaxPoolSize	Used to specify the maximum number of connections that can be opened in connection pool.
MinPoolSize	Used to specify the minimum number of connections that can be opened in the connection pool.
PoolId	Used to specify an ID for a connection pool.
Validate	Used for a connection to be validated when it is returned from the pool.

See Also

- [Pooling](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.2.10 Server Property

Serves to supply the server name for login.

Class

[TCustomDACConnection](#)

Syntax

```
property Server: string;
```

Remarks

Use the Server property to supply server name to handle server's request for a login.

See Also

- [Username](#)
- [Password](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.10.1.4.2.11 Username Property

Used to supply a user name for login.

Class

[TCustomDACConnection](#)

Syntax

```
property Username: string;
```

Remarks

Use the Username property to supply a user name to handle server's request for login. If this property is not set, IBDAC tries to connect with the user name.

Warning: Storing hard-coded user name and password entries as property values or in code for the OnLogin event handler can compromise server security.

See Also

- [Password](#)
- [Server](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3 Methods

Methods of the **TCustomDACConnection** class.

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

Public

Name	Description
ApplyUpdates	Overloaded. Applies changes in datasets.
Commit	Commits current transaction.

Connect	Establishes a connection to the server.
CreateSQL	Creates a component for queries execution.
Disconnect	Performs disconnect.
ExecProc	Allows to execute stored procedure or function providing its name and parameters.
ExecProcEx	Allows to execute a stored procedure or function.
ExecSQL	Executes a SQL statement with parameters.
ExecSQLEx	Executes any SQL statement outside the TQuery or TSQL components.
GetDatabaseNames	Returns a database list from the server.
GetKeyFieldNames	Provides a list of available key field names.
GetStoredProcNames	Returns a list of stored procedures from the server.
GetTableNames	Provides a list of available tables names.
MonitorMessage	Sends a specified message through the TCustomDASQLMonitor component.
Ping	Used to check state of connection to the server.
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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.1 ApplyUpdates Method

Applies changes in datasets.

Class

[TCustomDAConnection](#)

Overload List

Name	Description
ApplyUpdates	Applies changes from all active datasets.
ApplyUpdates(const DataSets: array of TCustomDADataSet)	Applies changes from the specified datasets.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Applies changes from the specified datasets.

Class

[TCustomDAConnection](#)

Syntax

```
procedure ApplyUpdates(const 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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.2 Commit Method

Commits current transaction.

Class

[TCustomDAConnection](#)

Syntax

```
procedure Commit; virtual;
```

Remarks

Call the Commit method to commit current transaction. On commit server writes permanently all pending data updates associated with the current transaction to the database and then ends the transaction. The current transaction is the last transaction started by calling StartTransaction.

See Also

- [Rollback](#)
- [StartTransaction](#)
- [TCustomIBCDataset.FetchAll](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.3 Connect Method

Establishes a connection to the server.

Class

[TCustomDAConnection](#)

Syntax

```
procedure Connect; overload; procedure Connect(const  
ConnectString: string); overload;
```

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](#)
- [TIBCConnection.Connected](#)
- [TIBCConnection.AssignConnect](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.4 CreateSQL Method

Creates a component for queries execution.

Class

[TCustomDAConnection](#)

Syntax

```
function CreateSQL: TCustomDASQL; virtual;
```

Return Value

A new instance of the class.

Remarks

Call the CreateSQL to return a new instance of the [TCustomDASQL](#) class and associates it with this connection object. In the descendant classes this method should be overridden to create an appropriate descendant of the TCustomDASQL component.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.5 Disconnect Method

Performs disconnect.

Class

[TCustomDAConnection](#)

Syntax

```
procedure Disconnect;
```

Remarks

Call the Disconnect method to drop a connection to database. Before the connection component is deactivated, all associated datasets are closed. Calling Disconnect is similar to setting the Connected property to False.

In most cases, closing a connection frees system resources allocated to the connection.

If user transaction is active, e.g. the [InTransaction](#) flag is set, calling to Disconnect will lead to applying the action specified in TIBCTransaction.DefaultCloseAction for the current user transaction.

Note: If a previously active connection is closed and then reopened, any associated datasets must be individually reopened; reopening the connection does not automatically reopen associated datasets.

See Also

- [Connect](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.6 ExecProc Method

Allows to execute stored procedure or function providing its name and parameters.

Class

[TCustomDAConnection](#)

Syntax

```
function ExecProc(const Name: string; const Params: array of
variant): variant; virtual;
```

Parameters

Name

Holds the name of the stored procedure or function.

Params

Holds the parameters of the stored procedure or function.

Return Value

the result of the stored procedure.

Remarks

Allows to execute stored procedure or function providing its name and parameters.

Use the following Name value syntax for executing specific overloaded routine:

"StoredProcName:1" or "StoredProcName:5". The first example executes the first overloaded stored procedure, while the second example executes the fifth overloaded procedure.

Assign parameters' values to the Params array in exactly the same order and number as they appear in the stored procedure declaration. Out parameters of the procedure can be accessed with the ParamByName procedure.

If the value of an input parameter was not included to the Params array, parameter default value is taken. Only parameters at the end of the list can be unincluded to the Params array. If the parameter has no default value, the NULL value is sent.

Note: Stored functions unlike stored procedures return result values that are obtained internally through the RESULT parameter. You will no longer have to provide anonymous value in the Params array to describe the result of the function. The stored function result is obtained from the Params[0] indexed property or with the ParamByName('RESULT') method call.

For further examples of parameter usage see [ExecSQL](#), [ExecSQLEx](#).

Example

For example, having stored function declaration presented in Example 1), you may execute it and retrieve its result with commands presented in Example 2):

```
Example 1)
CREATE procedure MY_SUM (
```

```
        A INTEGER,  
        B INTEGER)  
RETURNS (  
    RESULT INTEGER)  
as  
begin  
    Result = a + b;  
end;  
Example 2)  
Label1.Caption:= MyIBConnection1.ExecProc('My_Sum', [10, 20]);  
Label2.Caption:= MyIBConnection1.ParamByName('Result').AsString;
```

See Also

- [ExecProcEx](#)
- [ExecSQL](#)
- [ExecSQLEx](#)
- [TIBConnection.ParamByName](#)
- [TIBConnection.SQL](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.7 ExecProcEx Method

Allows to execute a stored procedure or function.

Class

[TCustomDAConnection](#)

Syntax

```
function ExecProcEx(const 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_value1
```

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.8 ExecSQL Method

Executes a SQL statement with parameters.

Class

[TCustomDAConnection](#)

Syntax

```
function ExecSQL(const Text: string): variant;  
overload;function ExecSQL(const 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.

Params array must contain all IN and OUT parameters defined in SQL statement. For OUT parameters provide any values of valid types so that they are explicitly defined before call to an ExecSQL method.

Out parameter with the name Result will hold the result of function having data type dtString. If none of the parameters in the Text statement is named Result, ExecSQL will return Null.

To get the values of OUT parameters use ParamByName function.

See Also

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

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.9 ExecSQLEx Method

Executes any SQL statement outside the TQuery or TSQL components.

Class

[TCustomDACConnection](#)

Syntax

```
function ExecSQLEx(const 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

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

See Also

- [ExecSQL](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.10 GetDatabaseNames Method

Returns a database list from the server.

Class

[TCustomDAConnection](#)

Syntax

```
procedure GetDatabaseNames(List: TStrings); virtual;
```

Parameters

List

A TStrings descendant that will be filled with database names.

Remarks

Populates a string list with the names of databases.

Note: Any contents already in the target string list object are eliminated and overwritten by data produced by GetDatabaseNames.

See Also

- [GetTableNames](#)
- [GetStoredProcNames](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.11 GetKeyFieldNames Method

Provides a list of available key field names.

Class

[TCustomDACConnection](#)

Syntax

```
procedure GetKeyFieldNames(const TableName: string; List: TStrings); virtual;
```

Parameters

TableName

Holds the table name

List

The list of available key field names

Return Value

Key field name

Remarks

Call the GetKeyFieldNames method to get the names of available key fields. Populates a string list with the names of key fields in tables.

See Also

- [GetTableNames](#)
- [GetStoredProcNames](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.12 GetStoredProcNames Method

Returns a list of stored procedures from the server.

Class

[TCustomDACConnection](#)

Syntax

```
procedure GetStoredProcNames(List: TStrings; AllProcs: boolean =  
False); virtual;
```

Parameters

List

A TStrings descendant that will be filled with the names of stored procedures in the database.

AllProcs

True, if stored procedures from all schemas or including system procedures (depending on the server) are returned. False otherwise.

Remarks

Call the GetStoredProcNames method to get the names of available stored procedures and functions. GetStoredProcNames populates a string list with the names of stored procs in the database. If AllProcs = True, the procedure returns to the List parameter the names of the stored procedures that belong to all schemas; otherwise, List will contain the names of functions that belong to the current schema.

Note: Any contents already in the target string list object are eliminated and overwritten by data produced by GetStoredProcNames.

See Also

- [GetDatabaseNames](#)
- [GetTableNames](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.13 GetTableNames Method

Provides a list of available tables names.

Class

[TCustomDACConnection](#)

Syntax

```
procedure GetTableNames(List: TStrings; AllTables: boolean =  
False; OnlyTables: boolean = False); virtual;
```

Parameters

List

A TStrings descendant that will be filled with table names.

AllTables

True, if procedure returns all table names including the names of system tables to the List parameter.

OnlyTables

Remarks

Call the GetTableNames method to get the names of available tables. Populates a string list with the names of tables in the database. If AllTables = True, procedure returns all table names including the names of system tables to the List parameter, otherwise List will not contain the names of system tables. If AllTables = True, the procedure returns to the List parameter the names of the tables that belong to all schemas; otherwise, List will contain the names of the tables that belong to the current schema.

Note: Any contents already in the target string list object are eliminated and overwritten by the data produced by GetTableNames.

See Also

- [GetDatabaseNames](#)
- [GetStoredProcNames](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.14 MonitorMessage Method

Sends a specified message through the [TCustomDASQLMonitor](#) component.

Class

[TCustomDAConnection](#)

Syntax

```
procedure MonitorMessage(const Msg: string);
```

Parameters

Msg

Message text that will be sent.

Remarks

Call the MonitorMessage method to output specified message via the [TCustomDASQLMonitor](#) component.

See Also

- [TCustomDASQLMonitor](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.15 Ping Method

Used to check state of connection to the server.

Class

[TCustomDAConnection](#)

Syntax

```
procedure Ping;
```

Remarks

The method is used for checking server connection state.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.16 RemoveFromPool Method

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

Class

[TCustomDAConnection](#)

Syntax

```
procedure RemoveFromPool;
```

Remarks

Call the `RemoveFromPool` method to mark the connection that should be deleted after disconnect instead of returning to the connection pool.

See Also

- [Pooling](#)
- [PoolingOptions](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.17 Rollback Method

Discards all current data changes and ends transaction.

Class

[TCustomDACConnection](#)

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.3.18 StartTransaction Method

Begins a new user transaction.

Class

[TCustomDACConnection](#)

Syntax

```
procedure StartTransaction; virtual;
```

Remarks

Call the StartTransaction method to begin a new user transaction against the database server. Before calling StartTransaction, an application should check the status of the [InTransaction](#) property. If InTransaction is True, indicating that a transaction is already in progress, a subsequent call to StartTransaction without first calling [Commit](#) or [Rollback](#) to end the current transaction raises EDatabaseError. Calling StartTransaction when connection is closed also raises EDatabaseError.

Updates, insertions, and deletions that take place after a call to StartTransaction are held by the server until an application calls Commit to save the changes, or Rollback to cancel them.

See Also

- [Commit](#)
- [Rollback](#)
- [InTransaction](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.4 Events

Events of the **TCustomDACConnection** class.

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

Public

Name	Description
OnConnectionLost	This event occurs when connection was lost.
OnError	This event occurs when an error has arisen in the connection.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.4.1 OnConnectionLost Event

This event occurs when connection was lost.

Class

[TCustomDACConnection](#)

Syntax

```
property OnConnectionLost: TConnectionLostEvent;
```

Remarks

Write the OnConnectionLost event handler to process fatal errors and perform failover.

Note: To use the OnConnectionLost event handler, you should explicitly add the MemData unit to the 'uses' list and set the TCustomDACConnection.Options.LocalFailover property to True.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.4.4.2 OnError Event

This event occurs when an error has arisen in the connection.

Class

[TCustomDAConnection](#)

Syntax

```
property OnError: TDAConnectionErrorEvent;
```

Remarks

Write the OnError event handler to respond to errors that arise with connection. Check the E parameter to get the error code. Set the Fail parameter to False to prevent an error dialog from being displayed and to raise the EAbort exception to cancel current operation. The default value of Fail is True.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5 TCustomDADataSet Class

Encapsulates general set of properties, events, and methods for working with data accessed through various database engines.

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

Unit

[DBAccess](#)

Syntax

```
TCustomDADataSet = class (TMemDataSet);
```

Remarks

TCustomDADataSet encapsulates general set of properties, events, and methods for working with data accessed through various database engines. All database-specific features are supported by descendants of TCustomDADataSet.

Applications should not use TCustomDADataSet objects directly.

Inheritance Hierarchy

[TMemDataSet](#)

TCustomDADataSet

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.1 Members

[TCustomDADataset](#) class overview.

Properties

Name	Description
BaseSQL	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions	Used to add WHERE conditions to a query
Connection	Used to specify a connection object to use to connect to a data store.
DataTypeMap	Used to set data type mapping rules
Debug	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
Disconnected	Used to keep dataset opened after connection is closed.
FetchRows	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL	Used to change the WHERE clause of SELECT statement and reopen a query.

FinalSQL	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
IsQuery	Used to check whether SQL statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
KeyFields	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
MacroCount	Used to get the number of macros associated with the Macros property.
Macros	Makes it possible to change SQL queries easily.
MasterFields	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
MasterSource	Used to specify the data source component which binds current dataset to the master one.
Options	Used to specify the behaviour of

	TCustomDADataset object.
ParamCheck	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
ParamCount	Used to indicate how many parameters are there in the Params property.
Params	Used to view and set parameter names, values, and data types dynamically.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
ReadOnly	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions	Used to indicate when the editing record is refreshed.
RowsAffected	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
SQL	Used to provide a SQL statement that a query component executes when its Open method is called.
SQLDelete	Used to specify a SQL statement that will be used when applying a deletion to a record.
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.
SQLRecCount	Used to specify the SQL statement that is used to get the record count when

	opening a dataset.
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.
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
AddWhere	Adds condition to the WHERE clause of SELECT statement in the SQL property.
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
BreakExec	Breaks execution of the SQL statement on the server.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.

CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
CreateBlobStream	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
DeleteWhere	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
Execute	Overloaded. Executes a SQL statement on the server.
Executing	Indicates whether SQL statement is still being executed.
Fetched	Used to find out whether TCustomDADataset has fetched all rows.
Fetching	Used to learn whether TCustomDADataset is still fetching rows.
FetchingAll	Used to learn whether TCustomDADataset is fetching all rows to the end.
FindKey	Searches for a record which contains specified field values.
FindMacro	Finds a macro with the specified name.
FindNearest	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.
FindParam	Determines if a parameter with the specified name exists in a dataset.
GetBlob (inherited from TMemDataSet)	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
GetDataType	Returns internal field types defined in the MemData and accompanying modules.
GetFieldObject	Returns a multireference shared object from field.
GetFieldPrecision	Retrieves the precision of a number field.
GetFieldScale	Retrieves the scale of a number field.
GetKeyFieldNames	Provides a list of available key field names.
GetOrderBy	Retrieves an ORDER BY clause from a SQL statement.
GotoCurrent	Sets the current record in this dataset similar to the current record in another dataset.
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.
Lock	Locks the current record.
MacroByName	Finds a macro with the specified name.
ParamByName	Sets or uses parameter information for a specific parameter based on its name.

Prepare	Allocates, opens, and parses cursor for a query.
RefreshRecord	Actualizes field values for the current record.
RestoreSQL	Restores the SQL property modified by AddWhere and SetOrderBy.
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.
SaveSQL	Saves the SQL property value to BaseSQL.
SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
SetOrderBy	Builds an ORDER BY clause of a SELECT statement.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
SQLSaved	Determines if the SQL property value was saved to the BaseSQL property.
UnLock	Releases a record lock.
UnPrepare (inherited from TMemDataSet)	Frees the resources allocated for a previously prepared query on the server and client sides.

UpdateResult (inherited from TMemDataSet)	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
UpdateStatus (inherited from TMemDataSet)	Indicates the current update status for the dataset when cached updates are enabled.

Events

Name	Description
AfterExecute	Occurs after a component has executed a query to database.
AfterFetch	Occurs after dataset finishes fetching data from server.
AfterUpdateExecute	Occurs after executing insert, delete, update, lock and refresh operations.
BeforeFetch	Occurs before dataset is going to fetch block of records from the server.
BeforeUpdateExecute	Occurs before executing insert, delete, update, lock, and refresh operations.
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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2 Properties

Properties of the **TCustomDADataset** class.

For a complete list of the **TCustomDADataset** class members, see the [TCustomDADataset](#)

[Members](#) topic.

Public

Name	Description
BaseSQL	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions	Used to add WHERE conditions to a query
Connection	Used to specify a connection object to use to connect to a data store.
DataTypeMap	Used to set data type mapping rules
Debug	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
Disconnected	Used to keep dataset opened after connection is closed.
FetchRows	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL	Used to change the WHERE clause of SELECT statement and reopen a query.
FinalSQL	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.

IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
IsQuery	Used to check whether SQL statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
KeyFields	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
MacroCount	Used to get the number of macros associated with the Macros property.
Macros	Makes it possible to change SQL queries easily.
MasterFields	Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.
MasterSource	Used to specify the data source component which binds current dataset to the master one.
Options	Used to specify the behaviour of TCustomDADataset object.
ParamCheck	Used to specify whether parameters for the Params property are generated automatically after the SQL

	property was changed.
ParamCount	Used to indicate how many parameters are there in the Params property.
Params	Used to view and set parameter names, values, and data types dynamically.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
ReadOnly	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions	Used to indicate when the editing record is refreshed.
RowsAffected	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
SQL	Used to provide a SQL statement that a query component executes when its Open method is called.
SQLDelete	Used to specify a SQL statement that will be used when applying a deletion to a record.
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.
SQLRecCount	Used to specify the SQL statement that is used to get the record count when opening a dataset.
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.
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.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.1 BaseSQL Property

Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.

Class

[TCustomDADataset](#)

Syntax

```
property BaseSQL : string;
```

Remarks

Use the BaseSQL property to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL, only macros are expanded. SQL text with all these changes can be returned by [FinalSQL](#).

See Also

- [FinalSQL](#)
- [AddWhere](#)
- [SaveSQL](#)
- [SQLSaved](#)
- [RestoreSQL](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.2 Conditions Property

Used to add WHERE conditions to a query

Class

[TCustomDADataset](#)

Syntax

```
property Conditions: TDAConditions stored False;
```

See Also

- [TDAConditions](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.3 Connection Property

Used to specify a connection object to use to connect to a data store.

Class

[TCustomDADataset](#)

Syntax

```
property Connection: TCustomDAConnection;
```

Remarks

Use the Connection property to specify a connection object that will be used to connect to a data store.

Set at design-time by selecting from the list of provided TCustomDACConnection or its descendant class objects.

At runtime, link an instance of a TCustomDACConnection descendant to the Connection property.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.4 DataTypeMap Property

Used to set data type mapping rules

Class

[TCustomDADataset](#)

Syntax

```
property DataTypeMap: TDAMapRules stored IsMapRulesStored;
```

See Also

- [TDAMapRules](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.5 Debug Property

Used to display the statement that is being executed and the values and types of its parameters.

Class

[TCustomDADataset](#)

Syntax

```
property Debug: boolean default False;
```

Remarks

Set the Debug property to True to display the statement that is being executed and the values and types of its parameters.

You should add the lbDacVcl unit to the uses clause of any unit in your project to make the Debug property work.

Note: If TIBCSQLMonitor is used in the project and the TIBCSQLMonitor.Active property is set to False, the debug window is not displayed.

See Also

- [TCustomDASQL.Debug](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.6 DetailFields Property

Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.

Class

[TCustomDADataset](#)

Syntax

```
property DetailFields: string;
```

Remarks

Use the DetailFields property to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship. DetailFields is a string containing one or more field names in the detail table. Separate field names with semicolons.

Use Field Link Designer to set the value in design time.

See Also

- [MasterFields](#)

- [MasterSource](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.7 Disconnected Property

Used to keep dataset opened after connection is closed.

Class

[TCustomDADataset](#)

Syntax

```
property Disconnected: boolean;
```

Remarks

Set the Disconnected property to True to keep dataset opened after connection is closed.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.8 FetchRows Property

Used to define the number of rows to be transferred across the network at the same time.

Class

[TCustomDADataset](#)

Syntax

```
property FetchRows: integer default 25;
```

Remarks

The number of rows that will be transferred across the network at the same time. This property can have a great impact on performance. So it is preferable to choose the optimal value of the FetchRows property for each SQL statement and software/hardware configuration experimentally.

The default value is 25.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.9 FilterSQL Property

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.

Note: the FilterSQL property adds a value to the WHERE condition as is. If you expect this value to be enclosed in brackets, you should bracket it explicitly.

Example

```
Query1.FilterSQL := 'Dept >= 20 and DName LIKE ''M%''';
```

See Also

- [AddWhere](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.10 FinalSQL Property

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.11 IsQuery Property

Used to check whether SQL statement returns rows.

Class

[TCustomDADataset](#)

Syntax

```
property IsQuery: boolean;
```

Remarks

After the TCustomDADataset component is prepared, the IsQuery property returns True if SQL statement is a SELECT query.

Use the IsQuery property to check whether the SQL statement returns rows or not.

IsQuery is a read-only property. Reading IsQuery on unprepared dataset raises an exception.

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.10.1.5.2.12 KeyFields Property

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

Class

[TCustomDADataset](#)

Syntax

```
property KeyFields: string;
```

Remarks

TCustomDADataset uses the KeyFields property to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database. For this feature KeyFields may hold a list of semicolon-delimited field names. If KeyFields is not defined before opening a dataset, TCustomDADataset requests information about primary keys from server sending an additional query.

KeyFields property may hold the name of a field which will be later assigned with an InterBase sequenced values. Beforehand InterBase generator must be created and its name passed to the [TCustomIBCDataset.KeyGenerator](#) property. Sequences are generated when either Insert or Post method is called. Which of these two methods is used to modify the database is determined by the [TCustomIBCDataset.GeneratorMode](#) property.

Note: Though keys may be created across a number of table fields, sequence is generated only for the first field found in the KeyFields property.

See Also

- [SQLDelete](#)
- [SQLInsert](#)
- [SQLRefresh](#)
- [SQLUpdate](#)
- [TCustomIBCDataset.KeyGenerator](#)

- [TCustomIBCDDataSet.GeneratorMode](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.13 MacroCount Property

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

Class

[TCustomDADDataSet](#)

Syntax

```
property MacroCount: word;
```

Remarks

Use the MacroCount property to get the number of macros associated with the Macros property.

See Also

- [Macros](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.14 Macros Property

Makes it possible to change SQL queries easily.

Class

[TCustomDADDataSet](#)

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

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

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.15 MasterFields Property

Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.

Class

[TCustomDADataset](#)

Syntax

```
property MasterFields: string;
```

Remarks

Use the MasterFields property after setting the [MasterSource](#) property to specify the names of one or more fields that are used as foreign keys for this dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.

MasterFields is a string containing one or more field names in the master table. Separate field names with semicolons.

Each time the current record in the master table changes, the new values in these fields are used to select corresponding records in this table for display.

Use Field Link Designer to set the values at design time after setting the MasterSource property.

See Also

- [DetailFields](#)
- [MasterSource](#)
- [Master/Detail Relationships](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.16 MasterSource Property

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

Class

[TCustomDADataset](#)

Syntax

```
property MasterSource: TDataSource;
```

Remarks

The MasterSource property specifies the data source component which binds current dataset to the master one.

TCustomDADataset uses MasterSource to extract foreign key fields values from the master dataset when building master/detail relationship between two datasets. MasterSource must point to another dataset; it cannot point to this dataset component.

When MasterSource is not **nil** dataset fills parameter values with corresponding field values from the current record of the master dataset.

Note: Do not set the DataSource property when building master/detail relationships. Although it points to the same object as the MasterSource property, it may lead to undesirable results.

See Also

- [MasterFields](#)
- [DetailFields](#)
- [Master/Detail Relationships](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.17 Options Property

Used to specify the behaviour of TCustomDADataset object.

Class

[TCustomDADataset](#)

Syntax

```
property Options: TDADatasetOptions;
```

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
AutoPrepare	Used to execute automatic Prepare on the query execution.
CacheCalcFields	Used to enable caching of the TField.Calculated and TField.Lookup fields.
CompressBlobMode	Used to store values of the BLOB fields in compressed form.
DefaultValues	Used to request default values/expressions from the server and assign them to the DefaultExpression property.
DetailDelay	Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.
FieldsOrigin	Used for TCustomDADataset to fill the Origin property of the TField objects by

	appropriate value when opening a dataset.
FlatBuffers	Used to control how a dataset treats data of the ftString and ftVarBytes fields.
InsertAllSetFields	Used to include all set dataset fields in the generated INSERT statement
LocalMasterDetail	Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.
LongStrings	Used to represent string fields with the length that is greater than 255 as TStringField.
MasterFieldsNullable	Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this option is enabled, the performance can get worse).
NumberRange	Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.
QueryRecCount	Used for TCustomDADataset to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.
QuoteNames	Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.
RemoveOnRefresh	Used for a dataset to locally remove a record that can not be found on the server.
RequiredFields	Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.
ReturnParams	Used to return the new value of fields to dataset after insert or update.
SetFieldsReadOnly	Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.
StrictUpdate	Used for TCustomDADataset to raise an exception when the number of updated or deleted records is not equal 1.
TrimFixedChar	Specifies whether to discard all trailing spaces in the string fields of a dataset.
UpdateAllFields	Used to include all dataset fields in the

	generated UPDATE and INSERT statements.
UpdateBatchSize	Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

See Also

- [Master/Detail Relationships](#)
- [TMemDataSet.CachedUpdates](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.18 ParamCheck Property

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

Class

[TCustomDADataset](#)

Syntax

```
property ParamCheck: boolean default True;
```

Remarks

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

Set ParamCheck to True to let dataset automatically generate the Params property for the dataset based on a SQL statement.

Setting ParamCheck to False can be used if the dataset component passes to a server the DDL statements that contain, for example, declarations of stored procedures which themselves will accept parameterized values. The default value is True.

See Also

- [Params](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.19 ParamCount Property

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

Class

[TCustomDADataset](#)

Syntax

```
property ParamCount: word;
```

Remarks

Use the ParamCount property to determine how many parameters are there in the Params property.

See Also

- [Params](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.20 Params Property

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

Class

[TCustomDADataset](#)

Syntax

```
property Params: TDAParams stored False;
```

Remarks

Contains the parameters for a query's SQL statement.

Access Params at runtime to view and set parameter names, values, and data types

dynamically (at design time use the Parameters editor to set the parameter information). Params is a zero-based array of parameter records. Index specifies the array element to access.

An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName.

See Also

- [ParamByName](#)
- [Macros](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.21 ReadOnly Property

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

Class

[TCustomDADataset](#)

Syntax

```
property ReadOnly: boolean default False;
```

Remarks

Use the ReadOnly property to prevent users from updating, inserting, or deleting data in the dataset. By default, ReadOnly is False, meaning that users can potentially alter data stored in the dataset.

To guarantee that users cannot modify or add data to a dataset, set ReadOnly to True.

When ReadOnly is True, the dataset's CanModify property is False.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.22 RefreshOptions Property

Used to indicate when the editing record is refreshed.

Class

[TCustomDADataset](#)

Syntax

```
property RefreshOptions: TRefreshOptions default [];
```

Remarks

Use the RefreshOptions property to determine when the editing record is refreshed.

Refresh is performed by the [RefreshRecord](#) method.

It queries the current record and replaces one in the dataset. Refresh record is useful when the table has triggers or the table fields have default values. Use roBeforeEdit to get actual data before editing.

The default value is [].

See Also

- [RefreshRecord](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.23 RowsAffected Property

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

Class

[TCustomDADataset](#)

Syntax

```
property RowsAffected: integer;
```

Remarks

Check RowsAffected to determine how many rows were inserted, updated, or deleted during the last query operation. If RowsAffected is -1, the query has not inserted, updated, or deleted any rows.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.24 SQL Property

Used to provide a SQL statement that a query component executes when its Open method is called.

Class

[TCustomDADataset](#)

Syntax

```
property SQL: TStrings;
```

Remarks

Use the SQL property to provide a SQL statement that a query component executes when its Open method is called. At the design time the SQL property can be edited by invoking the String List editor in Object Inspector.

When SQL is changed, TCustomDADataset calls Close and UnPrepare.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.25 SQLDelete Property

Used to specify a SQL statement that will be used when applying a deletion to a record.

Class

[TCustomDADataset](#)

Syntax

```
property SQLDelete: TStrings;
```

Remarks

Use the SQLDelete property to specify the SQL statement that will be used when applying a deletion to a record. Statements can be parameterized queries.

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

Example

```
DELETE FROM Orders  
WHERE  
    OrderID = :old_OrderID
```

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.26 SQLInsert Property

Used to specify the SQL statement that will be used when applying an insertion to a dataset.

Class

[TCustomDADataset](#)

Syntax

```
property SQLInsert: TStrings;
```

Remarks

Use the SQLInsert property to specify the SQL statement that will be used when applying an insertion to a dataset. Statements can be parameterized queries. Names of the parameters should be the same as field names. Parameters prefixed with OLD_ allow using current values of fields prior to the actual operation.

Use ReturnParam to return OUT parameters back to dataset.

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

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.27 SQLLock Property

Used to specify a SQL statement that will be used to perform a record lock.

Class

[TCustomDADataset](#)

Syntax

```
property SQLLock: TStrings;
```

Remarks

Use the SQLLock property to specify a SQL statement that will be used to perform a record lock. Statements can be 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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.28 SQLRecCount Property

Used to specify the SQL statement that is used to get the record count when opening a dataset.

Class

[TCustomDADataset](#)

Syntax

```
property SQLRecCount: TStrings;
```

Remarks

Use the SQLRecCount property to specify the SQL statement that is used to get the record count when opening a dataset. The SQL statement is used if the TDADatasetOptions.QueryRecCount property is True, and the TCustomDADataset.FetchAll property is False. Is not used if the FetchAll property is True.

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

See Also

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

- [SQLRefresh](#)
- [TDADatasetOptions](#)
- [FetchingAll](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.29 SQLRefresh Property

Used to specify a SQL statement that will be used to refresh current record by calling the [RefreshRecord](#) procedure.

Class

[TCustomDADataset](#)

Syntax

```
property SQLRefresh: TStrings;
```

Remarks

Use the SQLRefresh property to specify a SQL statement that will be used to refresh current record by calling the [RefreshRecord](#) procedure.

Different behavior is observed when the SQLRefresh property is assigned with a single WHERE clause that holds frequently altered search condition. In this case the WHERE clause from SQLRefresh is combined with the same clause of the SELECT statement in a SQL property and this final query is then sent to the database server.

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

Example

```
SELECT Shipname FROM Orders
WHERE
    OrderID = :OrderID
```

See Also

- [RefreshRecord](#)
- [SQL](#)

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.30 SQLUpdate Property

Used to specify a SQL statement that will be used when applying an update to a dataset.

Class

[TCustomDADataset](#)

Syntax

```
property SQLUpdate: TStrings;
```

Remarks

Use the SQLUpdate property to specify a SQL statement that will be used when applying an update to a dataset. Statements can be parameterized queries. Names of the parameters should be the same as field names. The parameters prefixed with OLD_ allow to use current values of fields prior to the actual operation.

Use ReturnParam to return OUT parameters back to the dataset.

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

Example

```
UPDATE Orders
  set
    ShipName = :ShipName
  WHERE
    OrderID = :Old_OrderID
```

See Also

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

- [SQLRefresh](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.2.31 UniDirectional Property

Used if an application does not need bidirectional access to records in the result set.

Class

[TCustomDADataset](#)

Syntax

```
property UniDirectional: boolean default False;
```

Remarks

Traditionally SQL cursors are unidirectional. They can travel only forward through a dataset. TCustomDADataset, however, permits bidirectional travelling by caching records. If an application does not need bidirectional access to the records in the result set, set UniDirectional to True. When UniDirectional is True, an application requires less memory and performance is improved. However, UniDirectional datasets cannot be modified. In FetchAll=False mode data is fetched on demand. When UniDirectional is set to True, data is fetched on demand as well, but obtained rows are not cached except for the current row. In case if the Unidirectional property is True, the FetchAll property will be automatically set to False. And if the FetchAll property is True, the Unidirectional property will be automatically set to False. The default value of UniDirectional is False, enabling forward and backward navigation.

Note: Pay attention to the specificity of using the FetchAll property=False

See Also

- [TIBCQuery.FetchAll](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3 Methods

Methods of the **TCustomDADataset** class.

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

Public

Name	Description
AddWhere	Adds condition to the WHERE clause of SELECT statement in the SQL property.
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
BreakExec	Breaks execution of the SQL statement on the server.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
CreateBlobStream	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
DeleteWhere	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.

EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
Execute	Overloaded. Executes a SQL statement on the server.
Executing	Indicates whether SQL statement is still being executed.
Fetched	Used to find out whether TCustomDADataset has fetched all rows.
Fetching	Used to learn whether TCustomDADataset is still fetching rows.
FetchingAll	Used to learn whether TCustomDADataset is fetching all rows to the end.
FindKey	Searches for a record which contains specified field values.
FindMacro	Finds a macro with the specified name.
FindNearest	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.
FindParam	Determines if a parameter with the specified name exists in a dataset.
GetBlob (inherited from TMemDataSet)	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
GetDataType	Returns internal field types defined in the MemData and accompanying modules.
GetFieldObject	Returns a multireference shared object from field.
GetFieldPrecision	Retrieves the precision of a number field.

GetFieldScale	Retrieves the scale of a number field.
GetKeyFieldNames	Provides a list of available key field names.
GetOrderBy	Retrieves an ORDER BY clause from a SQL statement.
GotoCurrent	Sets the current record in this dataset similar to the current record in another dataset.
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.
Lock	Locks the current record.
MacroByName	Finds a macro with the specified name.
ParamByName	Sets or uses parameter information for a specific parameter based on its name.
Prepare	Allocates, opens, and parses cursor for a query.
RefreshRecord	Actualizes field values for the current record.
RestoreSQL	Restores the SQL property modified by AddWhere and SetOrderBy.
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.
SaveSQL	Saves the SQL property value to BaseSQL.

SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
SetOrderBy	Builds an ORDER BY clause of a SELECT statement.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
SQLSaved	Determines if the SQL property value was saved to the BaseSQL property.
Unlock	Releases a record lock.
UnPrepare (inherited from TMemDataSet)	Frees the resources allocated for a previously prepared query on the server and client sides.
UpdateResult (inherited from TMemDataSet)	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
UpdateStatus (inherited from TMemDataSet)	Indicates the current update status for the dataset when cached updates are enabled.

See Also

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

Devart. All Rights Reserved.

5.10.1.5.3.1 AddWhere Method

Adds condition to the WHERE clause of SELECT statement in the SQL property.

Class

[TCustomDADataset](#)

Syntax

```
procedure AddWhere(const 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.

Note: the AddWhere method adds a value to the WHERE condition as is. If you expect this value to be enclosed in brackets, you should bracket it explicitly.

See Also

- [DeleteWhere](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.2 BreakExec Method

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. It makes sense to only call BreakExec from another thread.

See Also

- [TCustomDADataset.Execute](#)
- [TCustomDASQL.BreakExec](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.3 CreateBlobStream Method

Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.

Class

[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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.4 DeleteWhere Method

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.5 Execute Method

Executes a SQL statement on the server.

Class

[TCustomDADataset](#)

Overload List

Name	Description
Execute	Executes a SQL statement on the server.
Execute(Iters: integer; Offset: integer)	Used to perform Batch operations .

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Executes a SQL statement on the server.

Class

[TCustomDADataset](#)

Syntax

```
procedure Execute; overload; virtual;
```

Remarks

Call the Execute method to execute an SQL statement on the server. If SQL statement is a SELECT query, Execute calls the Open method.

Execute implicitly prepares SQL statement by calling the [TCustomDADataset.Prepare](#) method if the [TCustomDADataset.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

- [TCustomDADataset.AfterExecute](#)
- [TCustomDADataset.Executing](#)
- [TCustomDADataset.Prepare](#)
- [TIBCStoredProc.PrepareSQL](#)
- [TIBCStoredProc.Prepare](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Used to perform [Batch operations](#) .

Class

[TCustomDADataset](#)

Syntax

```
procedure Execute(Iter: integer; Offset: integer = 0); overload;  
virtual;
```

Parameters

Iter

Specifies the number of inserted rows.

Offset

Points the array element, which the Batch operation starts from. 0 by default.

Remarks

The Execute method executes the specified batch SQL query. See the [Batch operations](#) article for samples.

See Also

- [Batch operations](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.6 Executing Method

Indicates whether SQL statement is still being executed.

Class

[TCustomDADataset](#)

Syntax

```
function Executing: boolean;
```

Return Value

True, if SQL statement is still being executed.

Remarks

Check Executing to learn whether TCustomDADataset is still executing SQL statement.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.7 Fetched Method

Used to find out whether TCustomDADataset has fetched all rows.

Class

[TCustomDADataset](#)

Syntax

```
function Fetched: boolean; virtual;
```

Return Value

True, if all rows have been fetched.

Remarks

Call the Fetched method to find out whether TCustomDADataset has fetched all rows.

See Also

- [Fetching](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.8 Fetching Method

Used to learn whether TCustomDADataset is still fetching rows.

Class

[TCustomDADataset](#)

Syntax

```
function Fetching: boolean;
```

Return Value

True, if TCustomDADataset is still fetching rows.

Remarks

Check Fetching to learn whether TCustomDADataset is still fetching rows. Use the Fetching method if NonBlocking is True.

See Also

- [Executing](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.9 FetchingAll Method

Used to learn whether TCustomDADataset is fetching all rows to the end.

Class

[TCustomDADataset](#)

Syntax

```
function FetchingAll: boolean;
```

Return Value

True, if TCustomDADataset is fetching all rows to the end.

Remarks

Check FetchingAll to learn whether TCustomDADataset is fetching all rows to the end.

See Also

- [Executing](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.10 FindKey Method

Searches for a record which contains specified field values.

Class

[TCustomDADataset](#)

Syntax

```
function FindKey(const keyValues: array of System.TVarRec):  
Boolean;
```

Parameters

KeyValues

Holds a key.

Remarks

Call the FindKey method to search for a specific record in a dataset. KeyValues holds a comma-delimited array of field values, that is called a key.

This function is provided for BDE compatibility only. It is recommended to use functions [TMemDataSet.Locate](#) and [TMemDataSet.LocateEx](#) for the record search.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.11 FindMacro Method

Finds a macro with the specified name.

Class

[TCustomDADataset](#)

Syntax

```
function FindMacro(const value: string): TMacro;
```

Parameters

Value

Holds the name of a macro to search for.

Return Value

TMacro object if a match is found, nil otherwise.

Remarks

Call the FindMacro method to find a macro with the specified name. If a match is found, FindMacro returns the macro. Otherwise, it returns nil. Use this method instead of a direct reference to the [TMacros.Items](#) property to avoid depending on the order of the items.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.12 FindNearest Method

Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.

Class

[TCustomDADataset](#)

Syntax

```
procedure FindNearest(const KeyValues: array of System.TVarRec);
```

Parameters

KeyValues

Holds the values of the record key fields to which the cursor should be moved.

Remarks

Call the FindNearest method to move the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter. If there are no records that match or exceed the specified criteria, the cursor will not move.

This function is provided for BDE compatibility only. It is recommended to use functions [TMemDataSet.Locate](#) and [TMemDataSet.LocateEx](#) for the record search.

See Also

- [TMemDataSet.Locate](#)
- [TMemDataSet.LocateEx](#)
- [FindKey](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.13 FindParam Method

Determines if a parameter with the specified name exists in a dataset.

Class

[TCustomDADataset](#)

Syntax

```
function FindParam(const value: string): TDAParam;
```

Parameters

Value

Holds the name of the param for which to search.

Return Value

the TDAParam object for the specified Name. Otherwise it returns nil.

Remarks

Call the FindParam method to determine if a specified param component exists in a dataset. Name is the name of the param for which to search. If FindParam finds a param with a matching name, it returns a TDAParam object for the specified Name. Otherwise it returns nil.

See Also

- [Params](#)
- [ParamByName](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.14 GetDataType Method

Returns internal field types defined in the MemData and accompanying modules.

Class

[TCustomDADataset](#)

Syntax

```
function GetDataType(const FieldName: string): integer; virtual;
```

Parameters

FieldName

Holds the name of the field.

Return Value

internal field types defined in MemData and accompanying modules.

Remarks

Call the GetDataType method to return internal field types defined in the MemData and accompanying modules. Internal field data types extend the TFieldType type of VCL by specific database server data types. For example, ftString, ftFile, ftObject.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.15 GetFieldObject Method

Returns a multireference shared object from field.

Class

[TCustomDADataset](#)

Syntax

```
function GetFieldObject(Field: TField): TSharedObject;  
overload;function GetFieldObject(Field: TField; RecBuf:  
TRecordBuffer): TSharedObject; overload;function  
GetFieldObject(FieldDesc: TFieldDesc): TSharedObject;  
overload;function GetFieldObject(FieldDesc: TFieldDesc; RecBuf:  
TRecordBuffer): 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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.16 GetFieldPrecision Method

Retrieves the precision of a number field.

Class

[TCustomDADataset](#)

Syntax

```
function GetFieldPrecision(const FieldName: string): integer;
```

Parameters*FieldName*

Holds the existing field name.

Return Value

precision of number field.

Remarks

Call the GetFieldPrecision method to retrieve the precision of a number field. FieldName is the name of an existing field.

See Also

- [GetFieldScale](#)

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.10.1.5.3.17 GetFieldScale Method

Retrieves the scale of a number field.

Class

[TCustomDADataset](#)

Syntax

```
function GetFieldScale(const FieldName: string): integer;
```

Parameters

FieldName

Holds the existing field name.

Return Value

the scale of the number field.

Remarks

Call the GetFieldScale method to retrieve the scale of a number field. FieldName is the name of an existing field.

See Also

- [GetFieldPrecision](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.18 GetKeyFieldNames Method

Provides a list of available key field names.

Class

[TCustomDADataset](#)

Syntax

```
procedure GetKeyFieldNames(List: TStrings);
```

Parameters

List

The list of available key field names

Return Value

Key field name

Remarks

Call the `GetKeyFieldNames` method to get the names of available key fields. Populates a string list with the names of key fields in tables.

See Also

- [TCustomDACConnection.GetTableNames](#)
- [TCustomDACConnection.GetStoredProcNames](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.19 GetOrderBy Method

Retrieves an ORDER BY clause from a SQL statement.

Class

[TCustomDADataset](#)

Syntax

```
function GetOrderBy: string;
```

Return Value

an ORDER BY clause from the SQL statement.

Remarks

Call the `GetOrderBy` method to retrieve an ORDER BY clause from a SQL statement.

Note: `GetOrderBy` and `SetOrderBy` methods serve to process only quite simple queries and don't support, for example, subqueries.

See Also

- [SetOrderBy](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.20 GotoCurrent Method

Sets the current record in this dataset similar to the current record in another dataset.

Class

[TCustomDADataset](#)

Syntax

```
procedure GotoCurrent(DataSet: TCustomDADataset);
```

Parameters

DataSet

Holds the TCustomDADataset descendant to synchronize the record position with.

Remarks

Call the GotoCurrent method to set the current record in this dataset similar to the current record in another dataset. The key fields in both these DataSets must be coincident.

See Also

- [TMemDataSet.Locate](#)
- [TMemDataSet.LocateEx](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.21 Lock Method

Locks the current record.

Class

[TCustomDADataset](#)

Syntax

```
procedure Lock; virtual;
```

Remarks

Call the Lock method to lock the current record by executing the statement that is defined in the SQLLock property.

The Lock method sets the savepoint with the name LOCK_ + <component_name>.

See Also

- [UnLock](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.22 MacroByName Method

Finds a macro with the specified name.

Class

[TCustomDADataset](#)

Syntax

```
function MacroByName(const Value: string): TMacro;
```

Parameters

Value

Holds the name of a macro to search for.

Return Value

TMacro object if a match is found.

Remarks

Call the MacroByName method to find a macro with the specified name. If a match is found, MacroByName returns the macro. Otherwise, an exception is raised. Use this method instead of a direct reference to the [TMacros.Items](#) property to avoid depending on the order of the items.

To locate a parameter by name without raising an exception if the parameter is not found, use the FindMacro method.

To set a value to a macro, use the [TMacro.Value](#) property.

Example

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

See Also

- [TMacro](#)
- [Macros](#)
- [FindMacro](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.23 ParamByName Method

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

Class

[TCustomDADataset](#)

Syntax

```
function ParamByName(const Value: string): TDAParam;
```

Parameters

Value

Holds the name of the parameter for which to retrieve information.

Return Value

a TDAParam object.

Remarks

Call the ParamByName method to set or use parameter information for a specific parameter based on its name. Name is the name of the parameter for which to retrieve information. ParamByName is used to set a parameter's value at runtime and returns a [TDAParam](#) object.

Example

The following statement retrieves the current value of a parameter called "Contact" into an

edit box:

```
Edit1.Text := Query1.ParamsByName('Contact').AsString;
```

See Also

- [Params](#)
- [FindParam](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.24 Prepare Method

Allocates, opens, and parses cursor for a query.

Class

[TCustomDADataset](#)

Syntax

```
procedure Prepare; override;
```

Remarks

Call the Prepare method to allocate, open, and parse cursor for a query. Calling Prepare before executing a query improves application performance.

TCustomDADataset automatically prepares a query if it is executed without being prepared first. After execution, TCustomDADataset unprepares the query. When a query is executed a number of times, an application should always explicitly prepare the query to avoid multiple and unnecessary prepares and unprepares.

The UnPrepare method unprepares a query.

Note: When you change the text of a query at runtime, the query is automatically closed and unprepared.

See Also

- [TMemDataSet.Prepared](#)
- [TMemDataSet.UnPrepare](#)

- [Options](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.25 RefreshRecord Method

Actualizes field values for the current record.

Class

[TCustomDADataset](#)

Syntax

```
procedure RefreshRecord;
```

Remarks

Call the RefreshRecord method to actualize field values for the current record.

RefreshRecord performs query to database and refetches new field values from the returned cursor.

See Also

- [RefreshOptions](#)
- [SQLRefresh](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.26 RestoreSQL Method

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.27 SaveSQL Method

Saves the SQL property value to BaseSQL.

Class

[TCustomDADataset](#)

Syntax

```
procedure SaveSQL;
```

Remarks

Call the SaveSQL method to save the SQL property value to the BaseSQL property.

See Also

- [SQLSaved](#)
- [RestoreSQL](#)
- [BaseSQL](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.28 SetOrderBy Method

Builds an ORDER BY clause of a SELECT statement.

Class

[TCustomDADataset](#)

Syntax

```
procedure SetOrderBy(const 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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.29 SQLSaved Method

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

Class

[TCustomDADataset](#)

Syntax

```
function SQLSaved: boolean;
```

Return Value

True, if the SQL property value was saved to the BaseSQL property.

Remarks

Call the SQLSaved method to know whether the [SQL](#) property value was saved to the [BaseSQL](#) property.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.3.30 UnLock Method

Releases a record lock.

Class

[TCustomDADataset](#)

Syntax

```
procedure UnLock;
```

Remarks

Call the Unlock method to release the record lock made by the [Lock](#) method before.

Unlock is performed by rolling back to the savepoint set by the Lock method.

See Also

- [Lock](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.4 Events

Events of the **TCustomDADataset** class.

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

Public

Name	Description
AfterExecute	Occurs after a component has executed a query to database.
AfterFetch	Occurs after dataset finishes fetching data from server.
AfterUpdateExecute	Occurs after executing insert, delete, update, lock and refresh operations.
BeforeFetch	Occurs before dataset is going to fetch block of records from the server.
BeforeUpdateExecute	Occurs before executing insert, delete, update, lock, and refresh operations.
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.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.10.1.5.4.1 AfterExecute Event

Occurs after a component has executed a query to database.

Class

[TCustomDADataset](#)

Syntax

```
property AfterExecute: TAfterExecuteEvent;
```

Remarks

Occurs after a component has executed a query to database.

See Also

- [TCustomDADataset.Execute](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.4.2 AfterFetch Event

Occurs after dataset finishes fetching data from server.

Class

[TCustomDADataset](#)

Syntax

```
property AfterFetch: TAfterFetchEvent;
```

Remarks

The AfterFetch event occurs after dataset finishes fetching data from server.

See Also

- [BeforeFetch](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.4.3 AfterUpdateExecute Event

Occurs after executing insert, delete, update, lock and refresh operations.

Class

[TCustomDADataset](#)

Syntax

```
property AfterUpdateExecute: TUpdateExecuteEvent;
```

Remarks

Occurs after executing insert, delete, update, lock, and refresh operations. You can use AfterUpdateExecute to set the parameters of corresponding statements.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.4.4 BeforeFetch Event

Occurs before dataset is going to fetch block of records from the server.

Class

[TCustomDADataset](#)

Syntax

```
property BeforeFetch: TBeforeFetchEvent;
```

Remarks

The BeforeFetch event occurs every time before dataset is going to fetch a block of records from the server. Set Cancel to True to abort current fetch operation. To get an example on how to interrupt fetch operation refer to the Loader demo.

See Also

- [AfterFetch](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.5.4.5 BeforeUpdateExecute Event

Occurs before executing insert, delete, update, lock, and refresh operations.

Class

[TCustomDADataset](#)

Syntax

```
property BeforeUpdateExecute: TUpdateExecuteEvent;
```

Remarks

Occurs before executing insert, delete, update, lock, and refresh operations. You can use BeforeUpdateExecute to set the parameters of corresponding statements.

See Also

- [AfterUpdateExecute](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6 TCustomDASQL Class

A base class for components executing SQL statements that do not return result sets.

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

Unit

[DBAccess](#)

Syntax

```
TCustomDASQL = class(TComponent);
```

Remarks

TCustomDASQL is a base class that defines functionality for descendant classes which access database using SQL statements. Applications never use TCustomDASQL objects directly. Instead they use descendants of TCustomDASQL.

Use TCustomDASQL when client application must execute SQL statement or call stored procedure on the database server. The SQL statement should not retrieve rows from the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.1 Members

[TCustomDASQL](#) class overview.

Properties

Name	Description
ChangeCursor	Enables or disables changing screen cursor when executing commands in the NonBlocking mode.
Connection	Used to specify a connection object to use to connect to a data store.
Debug	Used to display the statement that is being executed and the values and types of its parameters.
FinalSQL	Used to return a SQL statement with expanded macros.
MacroCount	Used to get the number of macros associated with the Macros property.
Macros	Makes it possible to change SQL queries easily.
ParamCheck	Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.
ParamCount	Indicates the number of parameters in the Params property.
Params	Used to contain parameters for a SQL statement.
ParamValues	Used to get or set the values of individual field parameters that are identified by name.
Prepared	Used to indicate whether a query is prepared for execution.
RowsAffected	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

SQL	Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.
---------------------	---

Methods

Name	Description
BreakExec	Breaks execution of an SQL statement on the server.
Execute	Overloaded. Executes a SQL statement on the server.
Executing	Checks whether TCustomDASQL still executes a SQL statement.
FindMacro	Finds a macro with the specified name.
FindParam	Finds a parameter with the specified name.
MacroByName	Finds a macro with the specified name.
ParamByName	Finds a parameter with the specified name.
Prepare	Allocates, opens, and parses cursor for a query.
UnPrepare	Frees the resources allocated for a previously prepared query on the server and client sides.
WaitExecuting	Waits until TCustomDASQL executes a SQL statement.

Events

Name	Description
AfterExecute	Occurs after a SQL statement has been executed.

Devart. All Rights Reserved.

5.10.1.6.2 Properties

Properties of the **TCustomDASQL** class.

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

Public

Name	Description
ChangeCursor	Enables or disables changing screen cursor when executing commands in the NonBlocking mode.
Connection	Used to specify a connection object to use to connect to a data store.
Debug	Used to display the statement that is being executed and the values and types of its parameters.
FinalSQL	Used to return a SQL statement with expanded macros.
MacroCount	Used to get the number of macros associated with the Macros property.
Macros	Makes it possible to change SQL queries easily.
ParamCheck	Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.
ParamCount	Indicates the number of parameters in the Params property.
Params	Used to contain parameters for a SQL statement.
ParamValues	Used to get or set the values of individual field parameters that are

	identified by name.
Prepared	Used to indicate whether a query is prepared for execution.
RowsAffected	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
SQL	Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.1 ChangeCursor Property

Enables or disables changing screen cursor when executing commands in the NonBlocking mode.

Class

[TCustomDASQL](#)

Syntax

```
property changeCursor: boolean;
```

Remarks

Set the ChangeCursor property to False to prevent the screen cursor from changing to crSQLArrow when executing commands in the NonBlocking mode. The default value is True.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.2 Connection Property

Used to specify a connection object to use to connect to a data store.

Class

[TCustomDASQL](#)

Syntax

```
property Connection: TCustomDAConnection;
```

Remarks

Use the Connection property to specify a connection object that will be used to connect to a data store.

Set at design-time by selecting from the list of provided TCustomDAConnection or its descendant class objects.

At runtime, link an instance of a TCustomDAConnection descendant to the Connection property.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.3 Debug Property

Used to display the statement that is being executed and the values and types of its parameters.

Class

[TCustomDASQL](#)

Syntax

```
property Debug: boolean default False;
```

Remarks

Set the Debug property to True to display the statement that is being executed and the values and types of its parameters.

You should add the `lbDacVcl` unit to the `uses` clause of any unit in your project to make the `Debug` property work.

Note: If `TIBCSQLMonitor` is used in the project and the `TIBCSQLMonitor.Active` property is set to `False`, the debug window is not displayed.

See Also

- [TCustomDADataset.Debug](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.4 FinalSQL Property

Used to return a SQL statement with expanded macros.

Class

[TCustomDASQL](#)

Syntax

```
property FinalSQL: string;
```

Remarks

Read the `FinalSQL` property to return a SQL statement with expanded macros. This is the exact statement that will be passed on to the database server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.5 MacroCount Property

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

Class

[TCustomDASQL](#)

Syntax

```
property MacroCount: word;
```

Remarks

Use the MacroCount property to get the number of macros associated with the Macros property.

See Also

- [Macros](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.6 Macros Property

Makes it possible to change SQL queries easily.

Class

[TCustomDASQL](#)

Syntax

```
property Macros: TMacros stored False;
```

Remarks

With the help of macros you can easily change SQL query text at design- or runtime. Macros extend abilities of parameters and allow to change conditions in a WHERE clause or sort order in an ORDER BY clause. You just insert &MacroName in the SQL query text and change value of macro in the Macro property editor at design time or call the MacroByName function at run time. At the time of opening the query macro is replaced by its value.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.7 ParamCheck Property

Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.

Class

[TCustomDASQL](#)

Syntax

```
property ParamCheck: boolean default True;
```

Remarks

Use the ParamCheck property to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.

Set ParamCheck to True to let TCustomDASQL generate the Params property for the dataset based on a SQL statement automatically.

Setting ParamCheck to False can be used if the dataset component passes to a server the DDL statements that contain, for example, declarations of the stored procedures that will accept parameterized values themselves. The default value is True.

See Also

- [Params](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.8 ParamCount Property

Indicates the number of parameters in the Params property.

Class

[TCustomDASQL](#)

Syntax

```
property ParamCount: word;
```

Remarks

Use the ParamCount property to determine how many parameters are there in the Params property.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.9 Params Property

Used to contain parameters for a SQL statement.

Class

[TCustomDASQL](#)

Syntax

```
property Params: TDAParams stored False;
```

Remarks

Access the Params property at runtime to view and set parameter names, values, and data types dynamically (at design-time use the Parameters editor to set parameter properties).

Params is a zero-based array of parameter records. Index specifies the array element to access. An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName.

Example

Setting parameters at runtime:

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
with IBCSQL do  
  begin  
    SQL.Clear;  
    SQL.Add('INSERT INTO Temp_Table(Id, Name)');  
    SQL.Add('VALUES (:id, :Name)');  
    ParamByName('Id').AsInteger := 55;  
    Params[1].AsString := 'Green';  
    Execute;  
  end;  
end;
```

See Also

- [TDAParam](#)
- [FindParam](#)
- [Macros](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.10 ParamValues Property(Indexer)

Used to get or set the values of individual field parameters that are identified by name.

Class

[TCustomDASQL](#)

Syntax

```
property ParamValues [const 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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.11 Prepared Property

Used to indicate whether a query is prepared for execution.

Class

[TCustomDASQL](#)

Syntax

```
property Prepared: boolean;
```

Remarks

Check the Prepared property to determine if a query is already prepared for execution. True means that the query has already been prepared. As a rule prepared queries are executed faster, but the preparation itself also takes some time. One of the proper cases for using preparation is parametrized queries that are executed several times.

See Also

- [Prepare](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.12 RowsAffected Property

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

Class

[TCustomDASQL](#)

Syntax

```
property RowsAffected: integer;
```

Remarks

Check RowsAffected to determine how many rows were inserted, updated, or deleted during the last query operation. If RowsAffected is -1, the query has not inserted, updated, or deleted any rows.

Example

For correct initializing this property you should explicitly prepare SQL as it is shown in the example below.

```
IBCSQL.SQL.Text := 'Update Employee set salary = :sal where emp_no = :no';  
IBCSQL.Prepare;  
IBCSQL.Execute;  
AffRows := IBSQL.RowsAffected;  
IBCSQL.Unprepare;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.2.13 SQL Property

Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.

Class

[TCustomDASQL](#)

Syntax

```
property SQL: TStrings;
```

Remarks

Use the SQL property to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called. At design time the SQL property can be edited by invoking the String List editor in Object Inspector.

See Also

- [FinalSQL](#)
- [TCustomDASQL.Execute](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.3 Methods

Methods of the **TCustomDASQL** class.

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

Public

Name	Description
BreakExec	Breaks execution of an SQL statement on the server.
Execute	Overloaded. Executes a SQL statement on the server.
Executing	Checks whether TCustomDASQL still executes a SQL statement.
FindMacro	Finds a macro with the specified name.
FindParam	Finds a parameter with the specified name.
MacroByName	Finds a macro with the specified name.
ParamByName	Finds a parameter with the specified name.
Prepare	Allocates, opens, and parses cursor for a query.
UnPrepare	Frees the resources allocated for a previously prepared query on the server and client sides.
WaitExecuting	Waits until TCustomDASQL executes a SQL statement.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.3.1 BreakExec Method

Breaks execution of an SQL statement on the server.

Class

[TCustomDASQL](#)

Syntax

```
procedure BreakExec;
```

Remarks

Call the BreakExec method to break execution of an SQL statement on the server. It makes sense to call BreakExec only from another thread. Useful when NonBlocking is True.

See Also

- [TCustomDASQL.Execute](#)
- [TCustomDADataset.BreakExec](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.3.2 Execute Method

Executes a SQL statement on the server.

Class

[TCustomDASQL](#)

Overload List

Name	Description
Execute	Executes a SQL statement on the server.
Execute(Iters: integer; Offset: integer)	Used to perform Batch operations .

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Executes a SQL statement on the server.

Class

[TCustomDASQL](#)

Syntax

```
procedure Execute; overload; virtual;
```

Remarks

Call the Execute method to execute a SQL statement on the server. If the SQL statement has OUT parameters, use the [TCustomDASQL.ParamByName](#) method or the [TCustomDASQL.Params](#) property to get their values. Iters argument specifies the number of times this statement is executed for the DML array operations.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Used to perform [Batch operations](#) .

Class

[TCustomDASQL](#)

Syntax

```
procedure Execute(Iters: integer; Offset: integer = 0); overload;  
virtual;
```

Parameters

Iters

Specifies the number of inserted rows.

Offset

Points the array element, which the Batch operation starts from. 0 by default.

Remarks

The Execute method executes the specified batch SQL query. See the [Batch operations](#) article for samples.

See Also

- [Batch operations](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.3.3 Executing Method

Checks whether TCustomDASQL still executes a SQL statement.

Class

[TCustomDASQL](#)

Syntax

```
function Executing: boolean;
```

Return Value

True, if a SQL statement is still being executed by TCustomDASQL.

Remarks

Check Executing to find out whether TCustomDASQL still executes a SQL statement.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.3.4 FindMacro Method

Finds 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

TMacro object if a match is found, nil otherwise.

Remarks

Call the FindMacro method to find a macro with the specified name. If a match is found, FindMacro returns the macro. Otherwise, it returns nil. Use this method instead of a direct reference to the [TMacros.Items](#) property to avoid depending on the order of the items.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.3.5 FindParam Method

Finds a parameter with the specified name.

Class

[TCustomDASQL](#)

Syntax

```
function FindParam(const value: string): TDAParm;
```

Parameters

Value

Holds the parameter name to search for.

Return Value

a TDAParm object, if a parameter with the specified name has been found. If it has not, returns nil.

Remarks

Call the FindParam method to find a parameter with the specified name in a dataset.

See Also

- [ParamByName](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.3.6 MacroByName Method

Finds a macro with the specified name.

Class

[TCustomDASQL](#)

Syntax

```
function MacroByName(const Value: string): TMacro;
```

Parameters

Value

Holds the name of a macro to search for.

Return Value

TMacro object if a match is found.

Remarks

Call the MacroByName method to find a macro with the specified name. If a match is found, MacroByName returns the macro. Otherwise, an exception is raised. Use this method instead of a direct reference to the [TMacros.Items](#) property to avoid depending on the order of the items.

To locate a parameter by name without raising an exception if the parameter is not found, use the FindMacro method.

To set a value to a macro, use the [TMacro.Value](#) property.

See Also

- [TMacro](#)
- [Macros](#)
- [FindMacro](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.3.7 ParamByName Method

Finds a parameter with the specified name.

Class

[TCustomDASQL](#)

Syntax

```
function ParamByName(const Value: string): TDAParam;
```

Parameters

Value

Holds the name of the parameter to search for.

Return Value

a TDAParam object, if a match was found. Otherwise, an exception is raised.

Remarks

Use the ParamByName method to find a parameter with the specified name. If no parameter with the specified name found, an exception is raised.

Example

```
IBCSQL.Execute;  
Edit1.Text := IBCSQL.ParamsByName('Contact').AsString;
```

See Also

- [FindParam](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.3.8 Prepare Method

Allocates, opens, and parses cursor for a query.

Class

[TCustomDASQL](#)

Syntax

```
procedure Prepare; virtual;
```

Remarks

Call the Prepare method to allocate, open, and parse cursor for a query. Calling Prepare before executing a query improves application performance.

TCustomDADataset automatically prepares a query if it is executed without being prepared first. After execution, TCustomDADataset unprepares the query. When a query is executed a number of times, an application should always explicitly prepare the query to avoid multiple and unnecessary prepares and unprepares.

The UnPrepare method unprepares a query.

Note: When you change the text of a query at runtime, the query is automatically closed and unprepared.

See Also

- [Prepared](#)
- [UnPrepare](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.3.9 UnPrepare Method

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

Class

[TCustomDASQL](#)

Syntax

```
procedure UnPrepare; virtual;
```

Remarks

Call the UnPrepare method to free resources allocated for a previously prepared query on the server and client sides.

See Also

- [Prepare](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.3.10 WaitExecuting Method

Waits until TCustomDASQL executes a SQL statement.

Class

[TCustomDASQL](#)

Syntax

```
function waitExecuting(TimeOut: integer = 0): boolean;
```

Parameters

TimeOut

Holds the time in seconds to wait while TCustomDASQL executes the SQL statement. Zero means infinite time.

Return Value

True, if the execution of a SQL statement was completed in the preset time.

Remarks

Call the WaitExecuting method to wait until TCustomDASQL executes a SQL statement.

See Also

- [Executing](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.4 Events

Events of the **TCustomDASQL** class.

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

Public

Name	Description
AfterExecute	Occurs after a SQL statement has been executed.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.6.4.1 AfterExecute Event

Occurs after a SQL statement has been executed.

Class

[TCustomDASQL](#)

Syntax

```
property AfterExecute: TAfterExecuteEvent;
```

Remarks

Occurs after a SQL statement has been executed. This event may be used for descendant components which use multithreaded environment.

See Also

- [TCustomDASQL.Execute](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7 TCustomDAUpdateSQL Class

A base class for components that provide DML statements for more flexible control over data modifications.

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

Unit

[DBAccess](#)

Syntax

```
TCustomDAUpdateSQL = class(TComponent);
```

Remarks

TCustomDAUpdateSQL is a base class for components that provide DML statements for more flexible control over data modifications. Besides providing BDE compatibility, this component allows to associate a separate component for each update command.

See Also

- [TCustomIBDataSet.UpdateObject](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.1 Members

[TCustomDAUpdateSQL](#) class overview.

Properties

Name	Description
DataSet	Used to hold a reference to the TCustomDADataset object that is being updated.
DeleteObject	Provides ability to perform advanced adjustment of the delete operations.
DeleteSQL	Used when deleting a record.
InsertObject	Provides ability to perform advanced adjustment of insert operations.
InsertSQL	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
Apply	Sets parameters for a SQL statement and executes it to update a record.
ExecSQL	Executes a SQL statement.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2 Properties

Properties of the **TCustomDAUpdateSQL** class.

For a complete list of the **TCustomDAUpdateSQL** class members, see the

[TCustomDAUpdateSQL Members](#) topic.

Public

Name	Description
DataSet	Used to hold a reference to the TCustomDADataset object that is being updated.
SQL	Used to return a SQL statement for one of the ModifySQL, InsertSQL, or DeleteSQL properties.

Published

Name	Description
DeleteObject	Provides ability to perform advanced adjustment of the delete operations.
DeleteSQL	Used when deleting a record.
InsertObject	Provides ability to perform advanced adjustment of insert operations.
InsertSQL	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.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2.1 DataSet Property

Used to hold a reference to the TCustomDADataset object that is being updated.

Class

[TCustomDAUpdateSQL](#)

Syntax

```
property DataSet: TCustomDADataset;
```

Remarks

The DataSet property holds a reference to the TCustomDADataset object that is being updated. Generally it is not used directly.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2.2 DeleteObject Property

Provides ability to perform advanced adjustment of the delete operations.

Class

[TCustomDAUpdateSQL](#)

Syntax

```
property DeleteObject: TComponent;
```

Remarks

Assign SQL component or a TCustomIBCDataset descendant to this property to perform advanced adjustment of the delete operations. In some cases this can give some additional performance. Use the same principle to set the SQL property of an object as for setting the

[DeleteSQL](#) property.

See Also

- [DeleteSQL](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2.3 DeleteSQL Property

Used when deleting a record.

Class

[TCustomDAUpdateSQL](#)

Syntax

```
property DeleteSQL: TStrings;
```

Remarks

Set the DeleteSQL property to a DELETE statement to use when deleting a record. Statements can be parameterized queries with parameter names corresponding to the dataset field names.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2.4 InsertObject Property

Provides ability to perform advanced adjustment of insert operations.

Class

[TCustomDAUpdateSQL](#)

Syntax

```
property InsertObject: TComponent;
```

Remarks

Assign SQL component or TCustomIBCDataset descendant to this property to perform advanced adjustment of insert operations. In some cases this can give some additional performance. Set the SQL property of the object in the same way as used for the [InsertSQL](#) property.

See Also

- [InsertSQL](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2.5 InsertSQL Property

Used when inserting a record.

Class

[TCustomDAUpdatesQL](#)

Syntax

```
property InsertSQL: TStrings;
```

Remarks

Set the InsertSQL property to an INSERT INTO statement to use when inserting a record. Statements can be parameterized queries with parameter names corresponding to the dataset field names.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2.6 LockObject Property

Provides ability to perform advanced adjustment of lock operations.

Class

[TCustomDAUpdatesQL](#)

Syntax

```
property LockObject: TComponent;
```

Remarks

Assign a SQL component or TCustomIBCDataset descendant to this property to perform advanced adjustment of lock operations. In some cases that can give some additional performance. Set the SQL property of an object in the same way as used for the [LockSQL](#) property.

See Also

- [LockSQL](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2.7 LockSQL Property

Used to lock the current record.

Class

[TCustomDAUpdateSQL](#)

Syntax

```
property LockSQL: TStrings;
```

Remarks

Use the LockSQL property to lock the current record. Statements can be parameterized queries with parameter names corresponding to the dataset field names.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2.8 ModifyObject Property

Provides ability to perform advanced adjustment of modify operations.

Class

[TCustomDAUpdateSQL](#)

Syntax

```
property ModifyObject: TComponent;
```

Remarks

Assign a SQL component or TCustomIBCDataset descendant to this property to perform advanced adjustment of modify operations. In some cases this can give some additional performance. Set the SQL property of the object in the same way as used for the [ModifySQL](#) property.

See Also

- [ModifySQL](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2.9 ModifySQL Property

Used when updating a record.

Class

[TCustomDAUpdatesQL](#)

Syntax

```
property ModifySQL: TStrings;
```

Remarks

Set ModifySQL to an UPDATE statement to use when updating a record. Statements can be parameterized queries with parameter names corresponding to the dataset field names.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2.10 RefreshObject Property

Provides ability to perform advanced adjustment of refresh operations.

Class

[TCustomDAUpdatesQL](#)

Syntax

```
property RefreshObject: TComponent;
```

Remarks

Assign a SQL component or TCustomIBCDataset descendant to this property to perform advanced adjustment of refresh operations. In some cases that can give some additional performance. Set the SQL property of the object in the same way as used for the [RefreshSQL](#) property.

See Also

- [RefreshSQL](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2.11 RefreshSQL Property

Used to specify an SQL statement that will be used for refreshing the current record by [TCustomDADataset.RefreshRecord](#) procedure.

Class

[TCustomDAUpdatesQL](#)

Syntax

```
property RefreshSQL: TStrings;
```

Remarks

Use the RefreshSQL property to specify a SQL statement that will be used for refreshing the current record by the [TCustomDADataset.RefreshRecord](#) procedure.

You can assign to SQLRefresh a WHERE clause only. In such a case it is added to SELECT defined by the SQL property by [TCustomDADataset.AddWhere](#).

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

See Also

- [TCustomDADataSet.RefreshRecord](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.2.12 SQL Property(Indexer)

Used to return a SQL statement for one of the ModifySQL, InsertSQL, or DeleteSQL properties.

Class

[TCustomDAUpdateSQL](#)

Syntax

```
property SQL[UpdateKind: TUpdateKind]: TStrings;
```

Parameters

UpdateKind

Specifies which of update SQL statements to return.

Remarks

Returns a SQL statement for one of the ModifySQL, InsertSQL, or DeleteSQL properties, depending on the value of the UpdateKind index.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.3 Methods

Methods of the **TCustomDAUpdateSQL** class.

For a complete list of the **TCustomDAUpdateSQL** class members, see the

[TCustomDAUpdateSQL Members](#) topic.

Public

Name	Description
Apply	Sets parameters for a SQL

	statement and executes it to update a record.
ExecSQL	Executes a SQL statement.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.3.1 Apply Method

Sets parameters for a SQL statement and executes it to update a record.

Class

[TCustomDAUpdateSQL](#)

Syntax

```
procedure Apply(UpdateKind: TUpdateKind); virtual;
```

Parameters

UpdateKind

Specifies which of update SQL statements to execute.

Remarks

Call the Apply method to set parameters for a SQL statement and execute it to update a record. UpdateKind indicates which SQL statement to bind and execute.

Apply is primarily intended for manually executing update statements from an OnUpdateRecord event handler.

Note: If a SQL statement does not contain parameters, it is more efficient to call ExecSQL instead of Apply.

See Also

- [ExecSQL](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.7.3.2 ExecSQL Method

Executes a SQL statement.

Class

[TCustomDAUpdateSQL](#)

Syntax

```
procedure ExecSQL(UpdateKind: TUpdateKind);
```

Parameters

UpdateKind

Specifies the kind of update statement to be executed.

Remarks

Call the ExecSQL method to execute a SQL statement, necessary for updating the records belonging to a read-only result set when cached updates is enabled. UpdateKind specifies the statement to execute.

ExecSQL is primarily intended for manually executing update statements from the OnUpdateRecord event handler.

Note: To both bind parameters and execute a statement, call [Apply](#).

See Also

- [Apply](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.8 TDACondition Class

Represents a condition from the [TDAConditions](#) list.

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

Unit

[DBAccess](#)

Syntax

```
TDACondition = class(TCollectionItem);
```

Remarks

Manipulate conditions using [TDAConditions](#).

See Also

- [TDAConditions](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.8.1 Members

[TDACondition](#) class overview.

Properties

Name	Description
Enabled	Indicates whether the condition is enabled or not
Name	The name of the condition
Value	The value of the condition

Methods

Name	Description
Disable	Disables the condition
Enable	Enables the condition

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.8.2 Properties

Properties of the **TDACondition** class.

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

Published

Name	Description
Enabled	Indicates whether the condition is enabled or not
Name	The name of the condition
Value	The value of the condition

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.8.2.1 Enabled Property

Indicates whether the condition is enabled or not

Class

[TDACondition](#)

Syntax

```
property Enabled: Boolean default True;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.8.2.2 Name Property

The name of the condition

Class

[TDACondition](#)

Syntax

```
property Name: string;
```

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.8.2.3 Value Property

The value of the condition

Class

[TDACondition](#)

Syntax

```
property value: string;
```

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.8.3 Methods

Methods of the **TDACondition** class.

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

Public

Name	Description
Disable	Disables the condition
Enable	Enables the condition

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.8.3.1 Disable Method

Disables the condition

Class

[TDACondition](#)

Syntax

```
procedure Disable;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.8.3.2 Enable Method

Enables the condition

Class

[TDACondition](#)

Syntax

```
procedure Enable;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9 TDAConditions Class

Holds a collection of [TDACondition](#) objects.

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

Unit

[DBAccess](#)

Syntax

```
TDAConditions = class(TCollection);
```

Remarks

The given example code

```
UniTable1.Conditions.Add('1', 'JOB="MANAGER"');
UniTable1.Conditions.Add('2', 'SAL>2500');
UniTable1.Conditions.Enable;
UniTable1.Open;
```

will return the following SQL:

```
SELECT * FROM EMP
WHERE (JOB="MANAGER")
and
(SAL<2500)
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.1 Members

[TDAConditions](#) class overview.

Properties

Name	Description
Condition	Used to iterate through all the conditions.
Enabled	Indicates whether the condition is enabled
Items	Used to iterate through all conditions.
Text	The property returns condition names and values as CONDITION_NAME=CONDITION
WhereSQL	Returns the SQL WHERE condition added in the Conditions property.

Methods

Name	Description
Add	Overloaded. Adds a condition to the WHERE clause of the query.
Delete	Deletes the condition
Disable	Disables the condition
Enable	Enables the condition
Find	Search for TDACondition (the condition) by its name. If found, the TDACondition object is returned, otherwise - nil.
Get	Retrieving a TDACondition object by its name. If found, the TDACondition object is returned, otherwise - an exception is raised.
IndexOf	Retrieving condition index by its name. If found, this condition index is returned, otherwise - the method returns -1.
Remove	Removes the condition

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.2 Properties

Properties of the **TDAConditions** class.

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

Public

Name	Description
Condition	Used to iterate through all

	the conditions.
Enabled	Indicates whether the condition is enabled
Items	Used to iterate through all conditions.
Text	The property returns condition names and values as CONDITION_NAME=CONDITION
WhereSQL	Returns the SQL WHERE condition added in the Conditions property.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.2.1 Condition Property(Indexer)

Used to iterate through all the conditions.

Class

[TDAConditions](#)

Syntax

```
property Condition[Index: Integer]: TDACondition;
```

Parameters

Index

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.2.2 Enabled Property

Indicates whether the condition is enabled

Class

[TDAConditions](#)

Syntax

```
property Enabled: Boolean;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.2.3 Items Property(Indexer)

Used to iterate through all conditions.

Class

[TDAConditions](#)

Syntax

```
property Items[Index: Integer]: TDACondition; default;
```

Parameters

Index

Holds an index in the range 0..Count - 1.

Remarks

Use the Items property to iterate through all conditions. Index identifies the index in the range 0..Count - 1. Items can reference a particular condition by its index, but the [Condition](#) property is preferred in order to avoid depending on the order of the conditions.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.2.4 Text Property

The property returns condition names and values as CONDITION_NAME=CONDITION

Class

[TDAConditions](#)

Syntax

```
property Text: string;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.10.1.9.2.5 WhereSQL Property

Returns the SQL WHERE condition added in the Conditions property.

Class

[TDAConditions](#)

Syntax

```
property whereSQL: string;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.10.1.9.3 Methods

Methods of the **TDAConditions** class.

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

Public

Name	Description
Add	Overloaded. Adds a condition to the WHERE clause of the query.
Delete	Deletes the condition
Disable	Disables the condition
Enable	Enables the condition
Find	Search for TDACondition (the condition) by its name. If

	found, the TDACondition object is returned, otherwise - nil.
Get	Retrieving a TDACondition object by its name. If found, the TDACondition object is returned, otherwise - an exception is raised.
IndexOf	Retrieving condition index by its name. If found, this condition index is returned, otherwise - the method returns -1.
Remove	Removes the condition

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.3.1 Add Method

Adds a condition to the WHERE clause of the query.

Class

[TDAConditions](#)

Overload List

Name	Description
Add(const Value: string; Enabled: Boolean)	Adds a condition to the WHERE clause of the query.
Add(const Name: string; const Value: string; Enabled: Boolean)	Adds a condition to the WHERE clause of the query.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Adds a condition to the WHERE clause of the query.

Class

[TDAConditions](#)

Syntax

```
function Add(const value: string; Enabled: Boolean = True):  
TDACondition; overload;
```

Parameters

Value

The value of the condition

Enabled

Indicates that the condition is enabled

Remarks

If you want then to access the condition, you should use [Add](#) and its name in the Name parameter.

The given example code will return the following SQL:

```
SELECT * FROM EMP  
WHERE (JOB="MANAGER")  
and  
(SAL<2500)
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Adds a condition to the WHERE clause of the query.

Class

[TDAConditions](#)

Syntax

```
function Add(const Name: string; const value: string; Enabled:  
Boolean = True): TDACondition; overload;
```

Parameters

Name

Sets the name of the condition

Value

The value of the condition

Enabled

Indicates that the condition is enabled

Remarks

The given example code will return the following SQL:

```
SELECT * FROM EMP
WHERE (JOB="MANAGER")
and
(SAL<2500)
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.3.2 Delete Method

Deletes the condition

Class

[TDAConditions](#)

Syntax

```
procedure Delete(Index: integer);
```

Parameters

Index

Index of the condition

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.3.3 Disable Method

Disables the condition

Class

[TDAConditions](#)

Syntax

```
procedure Disable;
```

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.3.4 Enable Method

Enables the condition

Class

[TDAConditions](#)

Syntax

```
procedure Enable;
```

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.3.5 Find Method

Search for TDACondition (the condition) by its name. If found, the TDACondition object is returned, otherwise - nil.

Class

[TDAConditions](#)

Syntax

```
function Find(const Name: string): TDACondition;
```

Parameters

Name

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.3.6 Get Method

Retrieving a TDACondition object by its name. If found, the TDACondition object is returned, otherwise - an exception is raised.

Class

[TDAConditions](#)

Syntax

```
function Get(const Name: string): TDACondition;
```

Parameters

Name

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.3.7 IndexOf Method

Retrieving condition index by its name. If found, this condition index is returned, otherwise - the method returns -1.

Class

[TDAConditions](#)

Syntax

```
function IndexOf(const Name: string): Integer;
```

Parameters

Name

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.9.3.8 Remove Method

Removes the condition

Class

[TDAConditions](#)

Syntax

```
procedure Remove(const Name: string);
```

Parameters

Name

Specifies the name of the removed condition

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.10 TDACConnectionOptions Class

This class allows setting up the behaviour of the TDACConnection class.

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

Unit

[DBAccess](#)

Syntax

```
TDACConnectionOptions = class(TPersistent);
```

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.10.1 Members

[TDACConnectionOptions](#) class overview.

Properties

Name	Description
AllowImplicitConnect	Specifies whether to allow or not implicit connection opening.
DefaultSortType	Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the TMemDataSet.IndexFieldNames property of a dataset.
DisconnectedMode	Used to open a connection only when needed for

	performing a server call and closes after performing the operation.
KeepDesignConnected	Used to prevent an application from establishing a connection at the time of startup.
LocalFailover	If True, the TCustomDACConnection.OnConnectionLost event occurs and a failover operation can be performed after connection breaks.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.10.2 Properties

Properties of the **TDACConnectionOptions** class.

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

Public

Name	Description
DefaultSortType	Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the TMemDataSet.IndexFieldNames property of a dataset.
DisconnectedMode	Used to open a connection only when needed for performing a server call and closes after performing the operation.
KeepDesignConnected	Used to prevent an application from establishing a connection at the time of startup.

LocalFailover	If True, the TCustomDAConnection.OnConnectionLost event occurs and a failover operation can be performed after connection breaks.
-------------------------------	---

Published

Name	Description
AllowImplicitConnect	Specifies whether to allow or not implicit connection opening.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.10.2.1 Allow ImplicitConnect Property

Specifies whether to allow or not implicit connection opening.

Class

[TDAConnectionOptions](#)

Syntax

```
property AllowImplicitConnect: boolean default True;
```

Remarks

Use the AllowImplicitConnect property to specify whether allow or not implicit connection opening.

If a closed connection has AllowImplicitConnect set to True and a dataset that uses the connection is opened, the connection is opened implicitly to allow opening the dataset.

If a closed connection has AllowImplicitConnect set to False and a dataset that uses the

connection is opened, an exception is raised.

The default value is True.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.10.2.2 DefaultSortType Property

Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the [TMemDataSet.IndexFieldNames](#) property of a dataset.

Class

[TDAConnectionOptions](#)

Syntax

```
property DefaultSortType: TSortType default stCaseSensitive;
```

Remarks

Use the DefaultSortType property to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the [TMemDataSet.IndexFieldNames](#) property of a dataset.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.10.2.3 DisconnectedMode Property

Used to open a connection only when needed for performing a server call and closes after performing the operation.

Class

[TDAConnectionOptions](#)

Syntax

```
property DisconnectedMode: boolean default False;
```

Remarks

If True, connection opens only when needed for performing a server call and closes after performing the operation. Datasets remain opened when connection closes. May be useful to save server resources and operate in unstable or expensive network. Drawback of using disconnect mode is that each connection establishing requires some time for authorization. If connection is often closed and opened it can slow down the application work. See the [Disconnected Mode](#) topic for more information.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.10.2.4 KeepDesignConnected Property

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.10.2.5 LocalFailover Property

If True, the [TCustomDACconnection.OnConnectionLost](#) event occurs and a failover operation can be performed after connection breaks.

Class

[TDACconnectionOptions](#)

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.11 TDAConnectionSSLOptions Class

This class is used to set up the SSL options.

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

Unit

[DBAccess](#)

Syntax

```
TDAConnectionSSLOptions = class(TPersistent);
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.11.1 Members

[TDAConnectionSSLOptions](#) class overview.

Properties

Name	Description
CACert	Holds the path to the certificate authority file.
Cert	Holds the path to the client certificate.
CipherList	Holds the list of allowed SSL ciphers.

Key	Holds the path to the private client key.
---------------------	---

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.11.2 Properties

Properties of the **TDACConnectionSSLOptions** class.

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

Published

Name	Description
CACert	Holds the path to the certificate authority file.
Cert	Holds the path to the client certificate.
CipherList	Holds the list of allowed SSL ciphers.
Key	Holds the path to the private client key.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.11.2.1 CACert Property

Holds the path to the certificate authority file.

Class

[TDACConnectionSSLOptions](#)

Syntax

```
property CACert: string;
```

Remarks

Use the CACert property to specify the path to the certificate authority file.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.11.2.2 Cert Property

Holds the path to the client certificate.

Class

[TDAConnectionSSLOptions](#)

Syntax

```
property Cert: string;
```

Remarks

Use the Cert property to specify the path to the client certificate.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.11.2.3 CipherList Property

Holds the list of allowed SSL ciphers.

Class

[TDAConnectionSSLOptions](#)

Syntax

```
property CipherList: string;
```

Remarks

Use the CipherList property to specify the list of allowed SSL ciphers.

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.10.1.11.2.4 Key Property

Holds the path to the private client key.

Class

[TDAConnectionSSLOptions](#)

Syntax

```
property Key: string;
```

Remarks

Use the Key property to specify the path to the private client key.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12 TDADatasetOptions Class

This class allows setting up the behaviour of the TDADataset class.

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

Unit

[DBAccess](#)

Syntax

```
TDADatasetOptions = class(TPersistent);
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.1 Members

[TDADatasetOptions](#) class overview.

Properties

Name	Description
AutoPrepare	Used to execute automatic TCustomDADataset.Prepare on the query execution.
CacheCalcFields	Used to enable caching of the TField.Calculated and TField.Lookup fields.
CompressBlobMode	Used to store values of the BLOB fields in compressed form.
DefaultValues	Used to request default values/expressions from the server and assign them to the DefaultExpression property.
DetailDelay	Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.
FieldsOrigin	Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.
FlatBuffers	Used to control how a dataset treats data of the ftString and ftVarBytes fields.
InsertAllSetFields	Used to include all set dataset fields in the generated INSERT statement
LocalMasterDetail	Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.
LongStrings	Used to represent string fields with the length that is greater than 255 as TStringField.

MasterFieldsNullable	Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this option is enabled, the performance can get worse).
NumberRange	Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.
QueryRecCount	Used for TCustomDADataset to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.
QuoteNames	Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.
RemoveOnRefresh	Used for a dataset to locally remove a record that can not be found on the server.
RequiredFields	Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.
ReturnParams	Used to return the new value of fields to dataset after insert or update.
SetFieldsReadOnly	Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.
StrictUpdate	Used for TCustomDADataset to raise an exception when the

	number of updated or deleted records is not equal 1.
TrimFixedChar	Specifies whether to discard all trailing spaces in the string fields of a dataset.
UpdateAllFields	Used to include all dataset fields in the generated UPDATE and INSERT statements.
UpdateBatchSize	Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2 Properties

Properties of the **TDADatasetOptions** class.

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

Public

Name	Description
AutoPrepare	Used to execute automatic TCustomDADataset.Prepare on the query execution.
CacheCalcFields	Used to enable caching of the TField.Calculated and TField.Lookup fields.
CompressBlobMode	Used to store values of the BLOB fields in compressed form.
DefaultValues	Used to request default values/expressions from the server and assign them to the DefaultExpression property.

DetailDelay	Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.
FieldsOrigin	Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.
FlatBuffers	Used to control how a dataset treats data of the ftString and ftVarBytes fields.
InsertAllSetFields	Used to include all set dataset fields in the generated INSERT statement
LocalMasterDetail	Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.
LongStrings	Used to represent string fields with the length that is greater than 255 as TStringField.
MasterFieldsNullable	Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this option is enabled, the performance can get worse).
NumberRange	Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.
QueryRecCount	Used for TCustomDADataset to perform additional query to get the record count for this

	SELECT, so the RecordCount property reflects the actual number of records.
QuoteNames	Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.
RemoveOnRefresh	Used for a dataset to locally remove a record that can not be found on the server.
RequiredFields	Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.
ReturnParams	Used to return the new value of fields to dataset after insert or update.
SetFieldsReadOnly	Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.
StrictUpdate	Used for TCustomDADataset to raise an exception when the number of updated or deleted records is not equal 1.
TrimFixedChar	Specifies whether to discard all trailing spaces in the string fields of a dataset.
UpdateAllFields	Used to include all dataset fields in the generated UPDATE and INSERT statements.
UpdateBatchSize	Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.1 AutoPrepare Property

Used to execute automatic [TCustomDADataset.Prepare](#) on the query execution.

Class

[TDADatasetOptions](#)

Syntax

```
property AutoPrepare: boolean default False;
```

Remarks

Use the AutoPrepare property to execute automatic [TCustomDADataset.Prepare](#) on the query execution. Makes sense for cases when a query will be executed several times, for example, in Master/Detail relationships.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.2 CacheCalcFields Property

Used to enable caching of the TField.Calculated and TField.Lookup fields.

Class

[TDADatasetOptions](#)

Syntax

```
property CacheCalcFields: boolean default False;
```

Remarks

Use the CacheCalcFields property to enable caching of the TField.Calculated and TField.Lookup fields. It can be useful for reducing CPU usage for calculated fields. Using caching of calculated and lookup fields increases memory usage on the client side.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.3 CompressBlobMode Property

Used to store values of the BLOB fields in compressed form.

Class

[TDADatasetOptions](#)

Syntax

```
property CompressBlobMode: TCompressBlobMode default cbNone;
```

Remarks

Use the CompressBlobMode property to store values of the BLOB fields in compressed form. Add the MemData unit to uses list to use this option. Compression rate greatly depends on stored data, for example, usually graphic data compresses badly unlike text.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.4 DefaultValue Property

Used to request default values/expressions from the server and assign them to the DefaultValue property.

Class

[TDADatasetOptions](#)

Syntax

```
property DefaultValue: boolean default False;
```

Remarks

If True, the default values/expressions are requested from the server and assigned to the DefaultExpression property of TField objects replacing already existent values.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.5 DetailDelay Property

Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.

Class

[TDADatasetOptions](#)

Syntax

```
property DetailDelay: integer default 0;
```

Remarks

Use the DetailDelay property to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset. If DetailDelay is 0 (the default value) then refreshing of detail dataset occurs immediately. The DetailDelay option should be used for detail dataset.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.6 FieldsOrigin Property

Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.

Class

[TDADatasetOptions](#)

Syntax

```
property FieldsOrigin: boolean;
```

Remarks

If True, TCustomDADataset fills the Origin property of the TField objects by appropriate value when opening a dataset.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.7 FlatBuffers Property

Used to control how a dataset treats data of the ftString and ftVarBytes fields.

Class

[TDADatasetOptions](#)

Syntax

```
property FlatBuffers: boolean default False;
```

Remarks

Use the FlatBuffers property to control how a dataset treats data of the ftString and ftVarBytes fields. When set to True, all data fetched from the server is stored in record pdata without unused tails.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.8 InsertAllSetFields Property

Used to include all set dataset fields in the generated INSERT statement

Class

[TDADatasetOptions](#)

Syntax

```
property InsertAllSetFields: boolean default False;
```

Remarks

If True, all set dataset fields, including those set to NULL explicitly, will be included in the generated INSERT statements. Otherwise, only set fields containing not NULL values will be

included to the generated INSERT statement.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.9 LocalMasterDetail Property

Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.

Class

[TDatasetOptions](#)

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.10 LongStrings Property

Used to represent string fields with the length that is greater than 255 as TStringField.

Class

[TDatasetOptions](#)

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.11 MasterFieldsNullable Property

Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this option is enabled, the performance can get worse).

Class

[TDADatasetOptions](#)

Syntax

```
property MasterFieldsNullable: boolean default False;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.12 NumberRange Property

Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.

Class

[TDADatasetOptions](#)

Syntax

```
property NumberRange: boolean default False;
```

Remarks

Use the NumberRange property to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.13 QueryRecCount Property

Used for TCustomDADataset to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.

Class

[TDatasetOptions](#)

Syntax

```
property QueryRecCount: boolean default False;
```

Remarks

If True, and the 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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.14 QuoteNames Property

Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.

Class

[TDatasetOptions](#)

Syntax

```
property QuoteNames: boolean default False;
```

Remarks

If True, TCustomDADataset quotes all database object names in autogenerated SQL statements such as update SQL.

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.10.1.12.2.15 RemoveOnRefresh Property

Used for a dataset to locally remove a record that can not be found on the server.

Class

[TDADatasetOptions](#)

Syntax

```
property RemoveOnRefresh: boolean default True;
```

Remarks

When the RefreshRecord procedure can't find necessary record on the server and RemoveOnRefresh is set to True, dataset removes the record locally. Usually RefreshRecord can't find necessary record when someone else dropped the record or changed the key value of it.

This option makes sense only if the StrictUpdate option is set to False. If the StrictUpdate option is True, error will be generated regardless of the RemoveOnRefresh option value.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.16 RequiredFields Property

Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.

Class

[TDADatasetOptions](#)

Syntax

```
property RequiredFields: boolean default True;
```

Remarks

If True, TCustomDADataset sets the Required property of the TField objects for the NOT

NULL fields. It is useful when table has a trigger which updates the NOT NULL fields.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.17 ReturnParams Property

Used to return the new value of fields to dataset after insert or update.

Class

[TDADatasetOptions](#)

Syntax

```
property ReturnParams: boolean default False;
```

Remarks

Use the ReturnParams property to return the new value of fields to dataset after insert or update. The actual value of field after insert or update may be different from the value stored in the local memory if the table has a trigger. When ReturnParams is True, OUT parameters of the SQLInsert and SQLUpdate statements is assigned to the corresponding fields.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.18 SetFieldsReadOnly Property

Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.

Class

[TDADatasetOptions](#)

Syntax

```
property SetFieldsReadOnly: boolean default True;
```

Remarks

If True, dataset sets the ReadOnly property to True for all fields that do not belong to

UpdatingTable or can not be updated. Set this option for datasets that use automatic generation of the update SQL statements only.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.19 StrictUpdate Property

Used for TCustomDADataset to raise an exception when the number of updated or deleted records is not equal 1.

Class

[TDADatasetOptions](#)

Syntax

```
property strictUpdate: boolean default True;
```

Remarks

If True, TCustomDADataset raises an exception when the number of updated or deleted records is not equal 1. Setting this option also causes the exception if the RefreshRecord procedure returns more than one record. The exception does not occur when you execute SQL query, that doesn't return resultset.

Note: There can be problems if this option is set to True and triggers for UPDATE, DELETE, REFRESH commands that are defined for the table. So it is recommended to disable (set to False) this option with triggers.

TrimFixedChar specifies whether to discard all trailing spaces in the string fields of a dataset.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.20 TrimFixedChar Property

Specifies whether to discard all trailing spaces in the string fields of a dataset.

Class

[TDADatasetOptions](#)

Syntax

```
property TrimFixedChar: boolean default True;
```

Remarks

Specifies whether to discard all trailing spaces in the string fields of a dataset.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.21 UpdateAllFields Property

Used to include all dataset fields in the generated UPDATE and INSERT statements.

Class

[TDADatasetOptions](#)

Syntax

```
property UpdateAllFields: boolean default False;
```

Remarks

If True, all dataset fields will be included in the generated UPDATE and INSERT statements. Unspecified fields will have NULL value in the INSERT statements. Otherwise, only updated fields will be included to the generated update statements.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.12.2.22 UpdateBatchSize Property

Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

Class

[TDADatasetOptions](#)

Syntax

```
property UpdateBatchSize: Integer default 1;
```

Remarks

Use the UpdateBatchSize property to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch. Takes effect only when updating dataset in the [TMemDataSet.CachedUpdates](#) mode. The default value is 1.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.13 TDAEncryption Class

Used to specify the options of the data encryption in a dataset.

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

Unit

[DBAccess](#)

Syntax

```
TDAEncryption = class(TPersistent);
```

Remarks

Set the properties of Encryption to specify the options of the data encryption in a dataset.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.13.1 Members

[TDAEncryption](#) class overview.

Properties

Name	Description
Encryptor	Used to specify the encryptor class that will perform the data encryption.
Fields	Used to set field names for which encryption will be

performed.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.10.1.13.2 Properties

Properties of the **TDAEncryption** class.

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

Public

Name	Description
Encryptor	Used to specify the encryptor class that will perform the data encryption.

Published

Name	Description
Fields	Used to set field names for which encryption will be performed.

See Also

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

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.10.1.13.2.1 Encryptor Property

Used to specify the encryptor class that will perform the data encryption.

Class

[TDAEncryption](#)

Syntax

```
property Encryptor: TCREncryptor;
```

Remarks

Use the Encryptor property to specify the encryptor class that will perform the data encryption.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.13.2.2 Fields Property

Used to set field names for which encryption will be performed.

Class

[TDAEncryption](#)

Syntax

```
property Fields: string;
```

Remarks

Used to set field names for which encryption will be performed. Field names must be separated by semicolons.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14 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

TMapRule

TDAMapRule

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14.1 Members

[TDAMapRule](#) class overview.

Properties

Name	Description
DBLengthMax	Maximum DB field length, until which the rule is applied.
DBLengthMin	Minimum DB field length, starting from which the rule is applied.
DBScaleMax	Maximum DB field scale, until which the rule is applied to the specified DB field.
DBScaleMin	Minimum DB field Scale, starting from which the rule is applied to the specified DB field.
DBType	DB field type, that the rule is applied to.
FieldLength	The resultant field length in Delphi.
FieldName	DataSet field name, for which the rule is applied.
FieldScale	The resultant field Scale in Delphi.

FieldType	Delphi field type, that the specified DB type or DataSet field will be mapped to.
IgnoreErrors	Ignoring errors when converting data from DB to Delphi type.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14.2 Properties

Properties of the **TDAMapRule** class.

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

Published

Name	Description
DBLengthMax	Maximum DB field length, until which the rule is applied.
DBLengthMin	Minimum DB field length, starting from which the rule is applied.
DBScaleMax	Maximum DB field scale, until which the rule is applied to the specified DB field.
DBScaleMin	Minimum DB field Scale, starting from which the rule is applied to the specified DB field.
DBType	DB field type, that the rule is applied to.
FieldLength	The resultant field length in Delphi.
FieldName	DataSet field name, for which the rule is applied.
FieldScale	The resultant field Scale in Delphi.
FieldType	Delphi field type, that the

	specified DB type or DataSet field will be mapped to.
IgnoreErrors	Ignoring errors when converting data from DB to Delphi type.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14.2.1 DBLengthMax Property

Maximum DB field length, until which the rule is applied.

Class

[TDAMapRule](#)

Syntax

```
property DBLengthMax: Integer default r1Any;
```

Remarks

Setting maximum DB field length, until which the rule is applied to the specified DB field.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14.2.2 DBLengthMin Property

Minimum DB field length, starting from which the rule is applied.

Class

[TDAMapRule](#)

Syntax

```
property DBLengthMin: Integer default r1Any;
```

Remarks

Setting minimum DB field length, starting from which the rule is applied to the specified DB field.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14.2.3 DBScaleMax Property

Maximum DB field scale, until which the rule is applied to the specified DB field.

Class

[TDAMapRule](#)

Syntax

```
property DBScaleMax: Integer default r1Any;
```

Remarks

Setting maximum DB field scale, until which the rule is applied to the specified DB field.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14.2.4 DBScaleMin Property

Minimum DB field Scale, starting from which the rule is applied to the specified DB field.

Class

[TDAMapRule](#)

Syntax

```
property DBScaleMin: Integer default r1Any;
```

Remarks

Setting minimum DB field Scale, starting from which the rule is applied to the specified DB

field.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14.2.5 DBType Property

DB field type, that the rule is applied to.

Class

[TDAMapRule](#)

Syntax

```
property DBType: word default dtUnknown;
```

Remarks

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14.2.6 FieldLength Property

The resultant field length in Delphi.

Class

[TDAMapRule](#)

Syntax

```
property FieldLength: Integer default r1Any;
```

Remarks

Setting the Delphi field length after conversion.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14.2.7 FieldName Property

DataSet field name, for which the rule is applied.

Class

[TDAMapRule](#)

Syntax

```
property FieldName: string;
```

Remarks

Specifies the DataSet field name, that the rule is applied to. If the current rule is set for Connection, the rule will be applied to all fields with such name in DataSets related to this Connection.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14.2.8 FieldScale Property

The resultant field Scale in Delphi.

Class

[TDAMapRule](#)

Syntax

```
property FieldScale: Integer default r1Any;
```

Remarks

Setting the Delphi field Scale after conversion.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14.2.9 FieldType Property

Delphi field type, that the specified DB type or DataSet field will be mapped to.

Class

[TDAMapRule](#)

Syntax

```
property FieldType: TFieldType stored IsFieldTypeStored default  
ftUnknown;
```

Remarks

Setting Delphi field type, that the specified DB type or DataSet field will be mapped to.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.14.2.10 IgnoreErrors Property

Ignoring errors when converting data from DB to Delphi type.

Class

[TDAMapRule](#)

Syntax

```
property IgnoreErrors: Boolean default False;
```

Remarks

Allows to ignore errors while data conversion in case if data or DB data format cannot be recorded to the specified Delphi field type. The default value is false.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.15 TDAMapRules Class

Used for adding rules for DataSet fields mapping with both identifying by field name and by field type and Delphi field types.

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

Unit

[DBAccess](#)

Syntax

```
TDAMapRules = class(TMapRules);
```

Inheritance Hierarchy

TMapRules

TDAMapRules

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.15.1 Members

[TDAMapRules](#) class overview.

Properties

Name	Description
IgnoreInvalidRules	Used to avoid raising exception on mapping rules that can't be applied.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.15.2 Properties

Properties of the **TDAMapRules** class.

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

Published

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

[IgnoreInvalidRules](#)

Used to avoid raising exception on mapping rules that can't be applied.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.15.2.1 IgnoreInvalidRules Property

Used to avoid raising exception on mapping rules that can't be applied.

Class

[TDAMapRules](#)

Syntax

```
property IgnoreInvalidRules: boolean default False;
```

Remarks

Allows to ignore errors (not to raise exception) during data conversion in case if the data or DB data format cannot be recorded to the specified Delphi field type. The default value is false.

Note: In order to ignore errors occurring during data conversion, use the

[TDAMapRule.IgnoreErrors](#) property

See Also

- [TDAMapRule.IgnoreErrors](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.16 TDAMetaData Class

A class for retrieving metainformation of the specified database objects in the form of dataset.

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

Unit

[DBAccess](#)

Syntax

```
TDAMetaData = class(TMemDataSet);
```

Remarks

TDAMetaData is a TDataSet descendant standing for retrieving metainformation of the specified database objects in the form of dataset. First of all you need to specify which kind of metainformation you want to see. For this you need to assign the [TDAMetaData.MetaDataKind](#) property. Provide one or more conditions in the [TDAMetaData.Restrictions](#) property to diminish the size of the resultset and get only information you are interested in.

Use the [TDAMetaData.GetMetaDataKinds](#) method to get the full list of supported kinds of meta data. With the [TDAMetaData.GetRestrictions](#) method you can find out what restrictions are applicable to the specified MetaDataKind.

Example

The code below demonstrates how to get information about columns of the 'emp' table:

```
MetaData.Connection := Connection;  
MetaData.MetaDataKind := 'Columns';  
MetaData.Restrictions.Values['TABLE_NAME'] := 'Emp';  
MetaData.Open;
```

Inheritance Hierarchy

[TMemDataSet](#)

TDAMetaData

See Also

- [TDAMetaData.MetaDataKind](#)

- [TDAMetaData.Restrictions](#)
- [TDAMetaData.GetMetaDataKinds](#)
- [TDAMetaData.GetRestrictions](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.16.1 Members

[TDAMetaData](#) class overview.

Properties

Name	Description
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Connection	Used to specify a connection object to use to connect to a data store.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
MetaDataKind	Used to specify which kind of metainformation to show.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
Restrictions	Used to provide one or more conditions restricting the list

	of objects to be described.
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
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
GetBlob (inherited from TMemDataSet)	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.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
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
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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.16.2 Properties

Properties of the **TDAMetaData** class.

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

Public

Name	Description
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Connection	Used to specify a connection object to use to connect to a data store.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.

LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
MetaDataKind	Used to specify which kind of metainformation to show.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
Restrictions	Used to provide one or more conditions restricting the list of objects to be described.
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.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.16.2.1 Connection Property

Used to specify a connection object to use to connect to a data store.

Class

[TDAMetaData](#)

Syntax

```
property Connection: TCustomDAConnection;
```

Remarks

Use the Connection property to specify a connection object to use to connect to a data store.

Set at design-time by selecting from the list of provided TCustomDACConnection or its descendant class objects.

At runtime, set the Connection property to reference an instantiated TCustomDACConnection object.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.16.2.2 MetaDataKind Property

Used to specify which kind of metainformation to show.

Class

[TDAMetaData](#)

Syntax

```
property MetaDataKind: string;
```

Remarks

This string property specifies which kind of metainformation to show. The value of this property should be assigned before activating the component. If MetaDataKind equals to an empty string (the default value), the full value list that this property accepts will be shown.

They are described in the table below:

MetaDataKind	Description
Columns	show metainformation about columns of existing tables
Constraints	show metainformation about the constraints defined in the database
IndexColumns	show metainformation about indexed columns
Indexes	show metainformation about indexes in a database
MetaDataKinds	show the acceptable values of this property. You will get the same result if the MetadadataKind 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 restrictions are applicable to each MetaDataKind
Tables	show metainformation about existing tables
Databases	show metainformation about existing databases

If you provide a value that equals neither of the values described in the table, an error will be raised.

See Also

- [Restrictions](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.16.2.3 Restrictions Property

Used to provide one or more conditions restricting the list of objects to be described.

Class

[TDAMetaData](#)

Syntax

```
property Restrictions: TStrings;
```

Remarks

Use the Restriction list to provide one or more conditions restricting the list of objects to be described. To see the full list of restrictions and to which metadata kinds they are applicable, you should assign the Restrictions value to the MetaDataKind property and view the result.

See Also

- [MetaDataKind](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.16.3 Methods

Methods of the **TDAMetaData** class.

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

Public

Name	Description
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
GetBlob (inherited from TMemDataSet)	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.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
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.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.16.3.1 GetMetaDataKinds Method

Used to get values acceptable in the MetaDataKind property.

Class

[TDAMetaData](#)

Syntax

```
procedure GetMetaDataKinds(List: TStrings);
```

Parameters

List

Holds the object that will be filled with metadata kinds (restrictions).

Remarks

Call the GetMetaDataKinds method to get values acceptable in the MetaDataKind property. The List parameter will be cleared and then filled with values.

See Also

- [MetaDataKind](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.16.3.2 GetRestrictions Method

Used to find out which restrictions are applicable to a certain MetaDataKind.

Class

[TDAMetaData](#)

Syntax

```
procedure GetRestrictions(List: TStrings; const MetaDataKind:  
string);
```

Parameters

List

Holds the object that will be filled with metadata kinds (restrictions).

MetaDataKind

Holds the metadata kind for which restrictions are returned.

Remarks

Call the GetRestrictions method to find out which restrictions are applicable to a certain MetaDataKind. The List parameter will be cleared and then filled with values.

See Also

- [Restrictions](#)
- [GetMetaDataKinds](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17 TDAPParam Class

A class that forms objects to represent the values of the [parameters set](#).

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

Unit

[DBAccess](#)

Syntax

```
TDAParam = class(TParam);
```

Remarks

Use the properties of TDAParam to set the value of a parameter. Objects that use parameters create TDAParam objects to represent these parameters. For example, TDAParam objects are used by TCustomDASQL, TCustomDADataset.

TDAParam shares many properties with TField, as both describe the value of a field in a dataset. However, a TField object has several properties to describe the field binding and the way the field is displayed, edited, or calculated, that are not needed in a TDAParam object. Conversely, TDAParam includes properties that indicate how the field value is passed as a parameter.

See Also

- [TCustomDADataset](#)
- [TCustomDASQL](#)
- [TDAParams](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.1 Members

[TDAParam](#) class overview.

Properties

Name	Description
AsBlob	Used to set and read the value of the BLOB parameter as string.
AsBlobRef	Used to set and read the value of the BLOB parameter as a TBlob object.
AsFloat	Used to assign the value for a float field to a parameter.
AsInteger	Used to assign the value for an integer field to the

	parameter.
AsLargeInt	Used to assign the value for a LargeInteger field to the parameter.
AsMemo	Used to assign the value for a memo field to the parameter.
AsMemoRef	Used to set and read the value of the memo parameter as a TBlob object.
AsSQLTimeStamp	Used to specify the value of the parameter when it represents a SQL timestamp field.
AsString	Used to assign the string value to the parameter.
AsWideString	Used to assign the Unicode string value to the parameter.
DataType	Indicates the data type of the parameter.
IsNull	Used to indicate whether the value assigned to a parameter is NULL.
ParamType	Used to indicate the type of use for a parameter.
Size	Specifies the size of a string type parameter.
Value	Used to represent the value of the parameter as Variant.

Methods

Name	Description
AssignField	Assigns field name and field value to a param.
AssignFieldValue	Assigns the specified field properties and value to a parameter.
LoadFromFile	Places the content of a specified file into a TDAParam object.

LoadFromStream	Places the content from a stream into a TDAParam object.
SetBlobData	Overloaded. Writes the data from a specified buffer to BLOB.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2 Properties

Properties of the **TDAParam** class.

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

Public

Name	Description
AsBlob	Used to set and read the value of the BLOB parameter as string.
AsBlobRef	Used to set and read the value of the BLOB parameter as a TBlob object.
AsFloat	Used to assign the value for a float field to a parameter.
AsInteger	Used to assign the value for an integer field to the parameter.
AsLargeInt	Used to assign the value for a LargeInteger field to the parameter.
AsMemo	Used to assign the value for a memo field to the parameter.
AsMemoRef	Used to set and read the value of the memo parameter as a TBlob object.
AsSQLTimeStamp	Used to specify the value of the parameter when it represents a SQL

	timestamp field.
AsString	Used to assign the string value to the parameter.
AsWideString	Used to assign the Unicode string value to the parameter.
IsNull	Used to indicate whether the value assigned to a parameter is NULL.

Published

Name	Description
DataType	Indicates the data type of the parameter.
ParamType	Used to indicate the type of use for a parameter.
Size	Specifies the size of a string type parameter.
Value	Used to represent the value of the parameter as Variant.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.1 AsBlob Property

Used to set and read the value of the BLOB parameter as string.

Class

[TDAParam](#)

Syntax

```
property AsBlob: TBlobData;
```

Remarks

Use the AsBlob property to set and read the value of the BLOB parameter as string. Setting AsBlob will set the DataType property to ftBlob. AsBlob is the value of the parameter when it represents the value of LONG RAW type.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.2 AsBlobRef Property

Used to set and read the value of the BLOB parameter as a TBlob object.

Class

[TDAParam](#)

Syntax

```
property AsBlobRef: TBlob;
```

Remarks

Use the AsBlobRef property to set and read the value of the BLOB parameter as a TBlob object. Setting AsBlobRef will set the DataType property to ftBlob. Specifies the value of the parameter when it represents the value of LONG RAW type.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.3 AsFloat Property

Used to assign the value for a float field to a parameter.

Class

[TDAParam](#)

Syntax

```
property AsFloat: double;
```

Remarks

Use the AsFloat property to assign the value for a float field to the parameter. Setting AsFloat will set the DataType property to dtFloat.

Read the AsFloat property to determine the value that was assigned to an output parameter, represented as Double. The value of the parameter will be converted to the Double value if possible.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.4 AsInteger Property

Used to assign the value for an integer field to the parameter.

Class

[TDAParam](#)

Syntax

```
property AsInteger: LongInt;
```

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.5 AsLargeInt Property

Used to assign the value for a LargeInteger field to the parameter.

Class

[TDAParam](#)

Syntax

```
property AsLargeInt: Int64;
```

Remarks

Set the AsLargeInt property to assign the value for an Int64 field to the parameter. Setting AsLargeInt will set the DataType property to dtLargeint.

Read the AsLargeInt property to determine the value that was assigned to an output parameter, represented as a 64-bit integer. The value of the parameter will be converted to the Int64 value if possible.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.6 AsMemo Property

Used to assign the value for a memo field to the parameter.

Class

[TDAParam](#)

Syntax

```
property AsMemo: string;
```

Remarks

Use the AsMemo property to assign the value for a memo field to the parameter. Setting AsMemo will set the DataType property to ftMemo. AsMemo is the value of the parameter when it represents the value of LONG type.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.7 AsMemoRef Property

Used to set and read the value of the memo parameter as a TBlob object.

Class

[TDAParam](#)

Syntax

```
property AsMemoRef: TBlob;
```

Remarks

Use the AsMemoRef property to set and read the value of the memo parameter as a TBlob object. Setting AsMemoRef will set the DataType property to ftMemo. Specifies the value of the parameter when it represents the value of LONG type.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.8 AsSQLTimeStamp Property

Used to specify the value of the parameter when it represents a SQL timestamp field.

Class

[TDAParam](#)

Syntax

```
property AsSQLTimeStamp: TSQLTimeStamp;
```

Remarks

Set the AsSQLTimeStamp property to assign the value for a SQL timestamp field to the parameter. Setting AsSQLTimeStamp sets the DataType property to ftTimeStamp.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.9 AsString Property

Used to assign the string value to the parameter.

Class

[TDAParam](#)

Syntax

```
property AsString: string;
```

Remarks

Use the `AsString` property to assign the string value to the parameter. Setting `AsString` will set the `DataType` property to `ftString`.

Read the `AsString` property to determine the value that was assigned to an output parameter represented as a string. The value of the parameter will be converted to a string.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.10 `AsWideString` Property

Used to assign the Unicode string value to the parameter.

Class

[TDAParam](#)

Syntax

```
property AswideString: string;
```

Remarks

Set `AsWideString` to assign the Unicode string value to the parameter. Setting `AsWideString` will set the `DataType` property to `ftWideString`.

Read the `AsWideString` property to determine the value that was assigned to an output parameter, represented as a Unicode string. The value of the parameter will be converted to a Unicode string.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.11 `DataType` Property

Indicates the data type of the parameter.

Class

[TDAParam](#)

Syntax

```
property DataType: TFieldType stored IsDataTypeStored;
```

Remarks

DataType is set automatically when a value is assigned to a parameter. Do not set DataType for bound fields, as this may cause the assigned value to be misinterpreted.

Read DataType to learn the type of data that was assigned to the parameter. Every possible value of DataType corresponds to the type of a database field.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.12 IsNull Property

Used to indicate whether the value assigned to a parameter is NULL.

Class

[TDAParam](#)

Syntax

```
property IsNull: boolean;
```

Remarks

Use the IsNull property to indicate whether the value assigned to a parameter is NULL.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.13 ParamType Property

Used to indicate the type of use for a parameter.

Class

[TDAParam](#)

Syntax

```
property ParamType default DB . ptUnknown;
```

Remarks

Objects that use TDAParam objects to represent field parameters set ParamType to indicate the type of use for a parameter.

To learn the description of TParamType refer to Delphi Help.

Note: The value of ParamType is important for LONG, LONG RAW, BLOB and CLOB parameters. To write data to database, set ptInput to ParamType, to read data from database, set ptOutput to ParamType.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.14 Size Property

Specifies the size of a string type parameter.

Class

[TDAParam](#)

Syntax

```
property Size: integer default 0;
```

Remarks

Use the Size property to indicate the maximum number of characters the parameter may contain. Use the Size property only for Output parameters of the **ftString**, **ftFixedChar**, **ftBytes**, **ftVarBytes**, or **ftWideString** type.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.2.15 Value Property

Used to represent the value of the parameter as Variant.

Class

[TDAParam](#)

Syntax

```
property Value: variant stored IsValueStored;
```

Remarks

The Value property represents the value of the parameter as Variant.

Use Value in generic code that manipulates the values of parameters without the need to know the field type the parameter represent.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.3 Methods

Methods of the **TDAParam** class.

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

Public

Name	Description
AssignField	Assigns field name and field value to a param.
AssignFieldValue	Assigns the specified field properties and value to a parameter.
LoadFromFile	Places the content of a specified file into a TDAParam object.
LoadFromStream	Places the content from a stream into a TDAParam object.
SetBlobData	Overloaded. Writes the data from a specified buffer to BLOB.

See Also

- [TDAParam Class](#)

- [TDAParam Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.3.1 AssignField Method

Assigns field name and field value to a param.

Class

[TDAParam](#)

Syntax

```
procedure AssignField(Field: TField);
```

Parameters

Field

Holds the field which name and value should be assigned to the param.

Remarks

Call the AssignField method to assign field name and field value to a param.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.3.2 AssignFieldValue Method

Assigns the specified field properties and value to a parameter.

Class

[TDAParam](#)

Syntax

```
procedure AssignFieldValue(Field: TField; const value: variant);  
virtual;
```

Parameters

Field

Holds the field the properties of which will be assigned to the parameter.

Value

Holds the value for the parameter.

Remarks

Call the AssignFieldValue method to assign the specified field properties and value to a parameter.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.3.3 LoadFromFile Method

Places the content of a specified file into a TDAParam object.

Class

[TDAParam](#)

Syntax

```
procedure LoadFromFile(const FileName: string; BlobType: TBlobType);
```

Parameters

FileName

Holds the name of the file.

BlobType

Holds a value that modifies the DataType property so that this TDAParam object now holds the BLOB value.

Remarks

Use the LoadFromFile method to place the content of a file specified by FileName into a TDAParam object. The BlobType value modifies the DataType property so that this TDAParam object now holds the BLOB value.

See Also

- [LoadFromStream](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.3.4 LoadFromStream Method

Places the content from a stream into a TDAParam object.

Class

[TDAParam](#)

Syntax

```
procedure LoadFromStream(Stream: TStream; BlobType: TBlobType);  
virtual;
```

Parameters

Stream

Holds the stream to copy content from.

BlobType

Holds a value that modifies the DataType property so that this TDAParam object now holds the BLOB value.

Remarks

Call the LoadFromStream method to place the content from a stream into a TDAParam object. The BlobType value modifies the DataType property so that this TDAParam object now holds the BLOB value.

See Also

- [LoadFromFile](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.17.3.5 SetBlobData Method

Writes the data from a specified buffer to BLOB.

Class

[TDAParam](#)

Overload List

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

SetBlobData(Buffer: TValueBuffer)	Writes the data from a specified buffer to BLOB.
SetBlobData(Buffer: IntPtr; Size: Integer)	Writes the data from a specified buffer to BLOB.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Writes the data from a specified buffer to BLOB.

Class

[TDAParam](#)

Syntax

```
procedure SetBlobData(Buffer: IntPtr; Size: Integer); overload;
```

Parameters

Buffer

Holds the pointer to data.

Size

Holds the number of bytes to read from the buffer.

Remarks

Call the SetBlobData method to write data from a specified buffer to BLOB.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.18 TDAPParams Class

This class is used to manage a list of TDAPParam objects for an object that uses field parameters.

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

Unit

[DBAccess](#)

Syntax

```
TDAPParams = class(TParams);
```

Remarks

Use TDAPParams to manage a list of TDAPParam objects for an object that uses field parameters. For example, TCustomDADataset objects and TCustomDASQL objects use TDAPParams objects to create and access their parameters.

See Also

- [TCustomDADataset.Params](#)
- [TCustomDASQL.Params](#)
- [TDAPParam](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.18.1 Members

[TDAPParams](#) class overview.

Properties

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

Items	Used to iterate through all parameters.
-----------------------	---

Methods

Name	Description
FindParam	Searches for a parameter with the specified name.
ParamByName	Searches for a parameter with the specified name.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.18.2 Properties

Properties of the **TDAParams** class.

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

Public

Name	Description
Items	Used to iterate through all parameters.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.18.2.1 Items Property(Indexer)

Used to iterate through all parameters.

Class

[TDAParams](#)

Syntax

```
property Items[Index: integer]: TDAParam; default;
```

Parameters

Index

Holds an index in the range 0..Count - 1.

Remarks

Use the Items property to iterate through all parameters. Index identifies the index in the range 0..Count - 1. Items can reference a particular parameter by its index, but the ParamByName method is preferred in order to avoid depending on the order of the parameters.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.18.3 Methods

Methods of the **TDAParams** class.

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

Public

Name	Description
FindParam	Searches for a parameter with the specified name.
ParamByName	Searches for a parameter with the specified name.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.18.3.1 FindParam Method

Searches for a parameter with the specified name.

Class

[TDAParams](#)

Syntax

```
function FindParam(const value: string): TDAParam;
```

Parameters

Value

Holds the parameter name.

Return Value

a parameter, if a match was found. Nil otherwise.

Remarks

Use the FindParam method to find a parameter with the name passed in Value. If a match is found, FindParam returns the parameter. Otherwise, it returns nil. Use this method rather than a direct reference to the Items property to avoid depending on the order of the entries.

To locate more than one parameter at a time by name, use the GetParamList method instead. To get only the value of a named parameter, use the ParamValues property.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.18.3.2 ParamByName Method

Searches for a parameter with the specified name.

Class

[TDAParams](#)

Syntax

```
function ParamByName(const value: string): TDAParam;
```

Parameters

Value

Holds the parameter name.

Return Value

a parameter, if the match was found. otherwise an exception is raised.

Remarks

Use the ParamByName method to find a parameter with the name passed in Value. If a match was found, ParamByName returns the parameter. Otherwise, an exception is raised. Use this method rather than a direct reference to the [Items](#) property to avoid depending on the order of the entries.

To locate a parameter by name without raising an exception if the parameter is not found, use the FindParam method.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19 TDATransaction Class

A base class that implements functionality for controlling transactions.

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

Unit

[DBAccess](#)

Syntax

```
TDATransaction = class(TComponent);
```

Remarks

TDATransaction is a base class for components implementing functionality for managing transactions.

Do not create instances of TDATransaction. Use descendants of the TDATransaction class instead.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19.1 Members

[TDATransaction](#) class overview.

Properties

Name	Description
Active	Used to determine if the transaction is active.
DefaultCloseAction	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
Commit	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
OnCommit	Occurs after the transaction has been successfully committed.
OnCommitRetaining	Occurs after CommitRetaining has been executed.
OnError	Used to process errors that occur during executing a transaction.
OnRollback	Occurs after the transaction has been successfully rolled back.

OnRollbackRetaining	Occurs after RollbackRetaining has been executed.
-------------------------------------	---

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19.2 Properties

Properties of the **TDATransaction** class.

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

Public

Name	Description
Active	Used to determine if the transaction is active.
DefaultCloseAction	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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19.2.1 Active Property

Used to determine if the transaction is active.

Class

[TDATransaction](#)

Syntax

```
property Active: boolean;
```

Remarks

Indicates whether the transaction is active. This property is read-only.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19.2.2 DefaultCloseAction Property

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

Class

[TDATransaction](#)

Syntax

```
property DefaultCloseAction: TCRTransactionAction default  
taRollback;
```

Remarks

Use DefaultCloseAction to specify the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19.3 Methods

Methods of the **TDATransaction** class.

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

Public

Name	Description
Commit	Commits the current transaction.

Rollback	Discards all modifications of data associated with the current transaction and ends the transaction.
StartTransaction	Begins a new transaction.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19.3.1 Commit Method

Commits the current transaction.

Class

[TDATransaction](#)

Syntax

```
procedure Commit; virtual;
```

Remarks

Call the Commit method to commit the current transaction. On commit server writes permanently all pending data updates associated with the current transaction to the database, and then finishes the transaction.

See Also

- [Rollback](#)
- [StartTransaction](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19.3.2 Rollback Method

Discards all modifications of data associated with the current transaction and ends the transaction.

Class

[TDATransaction](#)

Syntax

```
procedure Rollback; virtual;
```

Remarks

Call Rollback to cancel all data modifications made within the current transaction to the database server, and finish the transaction.

See Also

- [Commit](#)
- [StartTransaction](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19.3.3 StartTransaction Method

Begins a new transaction.

Class

[TDATransaction](#)

Syntax

```
procedure StartTransaction; virtual;
```

Remarks

Call the StartTransaction method to begin a new transaction against the database server. Before calling StartTransaction, an application should check the [Active](#) property. If TDATransaction.Active is True, indicating that a transaction is already in progress, a

subsequent call to `StartTransaction` will raise `EDatabaseError`. An active transaction must be finished by call to [Commit](#) or [Rollback](#) before call to `StartTransaction`. Call to `StartTransaction` when connection is closed also will raise `EDatabaseError`.

Updates, insertions, and deletions that take place after a call to `StartTransaction` are held by the server until the application calls [Commit](#) to save the changes, or [Rollback](#) to cancel them.

See Also

- [Commit](#)
- [Rollback](#)
- [TIBCTransaction.Active](#)
- [TIBCTransaction.Commit](#)
- [TIBCTransaction.Rollback](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19.4 Events

Events of the **TDATransaction** class.

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

Public

Name	Description
OnCommit	Occurs after the transaction has been successfully committed.
OnCommitRetaining	Occurs after <code>CommitRetaining</code> has been executed.
OnError	Used to process errors that occur during executing a transaction.
OnRollback	Occurs after the transaction has been successfully rolled back.

[OnRollbackRetaining](#)

Occurs after RollbackRetaining has been executed.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.10.1.19.4.1 OnCommit Event

Occurs after the transaction has been successfully committed.

Class

[TDATransaction](#)

Syntax

```
property OnCommit: TNotifyEvent;
```

Remarks

The OnCommit event fires when the M:Devart.Dac.TDATransaction.Commit method is executed, just after the transaction is successfully committed. In order to respond to the [TIBCTransaction.CommitRetaining](#) method execution, the [OnCommitRetaining](#) event is used. When an error occurs during commit, the [OnError](#) event fires.

See Also

- [Commit](#)
- [TIBCTransaction.CommitRetaining](#)
- [OnCommitRetaining](#)
- [OnError](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.10.1.19.4.2 OnCommitRetaining Event

Occurs after CommitRetaining has been executed.

Class

[TDATransaction](#)

Syntax

```
property OnCommitRetaining: TNotifyEvent;
```

Remarks

The OnCommitRetaining event fires when the CommitRetaining method is executed, just after the transaction is successfully committed. In order to respond to the M:Devart.Dac.TDATransaction.Commit method execution, the [OnCommit](#) event is used.

When an error occurs during commit, the [OnError](#) event fired.

See Also

- [TIBCTransaction.CommitRetaining](#)
- [Commit](#)
- [OnCommit](#)
- [OnError](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19.4.3 OnError Event

Used to process errors that occur during executing a transaction.

Class

[TDATransaction](#)

Syntax

```
property OnError: TDATransactionErrorEvent;
```

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19.4.4 OnRollback Event

Occurs after the transaction has been successfully rolled back.

Class

[TDATransaction](#)

Syntax

```
property OnRollback: TNotifyEvent;
```

Remarks

The OnRollback event fires when the M:Devart.Dac.TDATransaction.Rollback method is executed, just after the transaction is successfully rolled back. In order to respond to the [TIBCTransaction.RollbackRetaining](#) method execution, the [OnRollbackRetaining](#) event is used.

When an error occurs during rollback, the [OnError](#) event fired.

See Also

- [Rollback](#)
- [TIBCTransaction.RollbackRetaining](#)
- [OnRollbackRetaining](#)
- [OnError](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.19.4.5 OnRollbackRetaining Event

Occurs after RollbackRetaining has been executed.

Class

[TDATransaction](#)

Syntax

```
property OnRollbackRetaining: TNotifyEvent;
```

Remarks

The OnRollbackRetaining event fires when the RollbackRetaining method is executed, just after the transaction is successfully rolled back. In order to respond to the [Rollback](#) method execution, the [OnRollback](#) event is used. When an error occurs during rollback, the [OnError](#) event fired.

See Also

- [Rollback](#)
- [TIBCTransaction.RollbackRetaining](#)
- [OnRollback](#)
- [OnError](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.20 TMacro Class

Object that represents the value of a macro.

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

Unit

[DBAccess](#)

Syntax

```
TMacro = class(TCollectionItem);
```

Remarks

TMacro object represents the value of a macro. Macro is a variable that holds string value. You just insert **&** MacroName in a SQL query text and change the value of macro by the Macro property editor at design time or the Value property at run time. At the time of opening query macro is replaced by its value.

If by any reason it is not convenient for you to use the ' **&** ' symbol as a character of macro replacement, change the value of the MacroChar variable.

See Also

- [TMacros](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.20.1 Members

[TMacro](#) class overview.

Properties

Name	Description
Active	Used to determine if the macro should be expanded.
AsDateTime	Used to set the TDateTime value to a macro.
AsFloat	Used to set the float value to a macro.
AsInteger	Used to set the integer value to a macro.
AsString	Used to assign the string value to a macro.
Name	Used to identify a particular macro.
Value	Used to set the value to a macro.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.20.2 Properties

Properties of the **TMacro** class.

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

Public

Name	Description
AsDateTime	Used to set the TDateTime value to a macro.
AsFloat	Used to set the float value to a macro.
AsInteger	Used to set the integer value to a macro.
AsString	Used to assign the string value to a macro.

Published

Name	Description
Active	Used to determine if the macro should be expanded.
Name	Used to identify a particular macro.
Value	Used to set the value to a macro.

See Also

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

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.20.2.1 Active Property

Used to determine if the macro should be expanded.

Class

[TMacro](#)

Syntax

```
property Active: boolean default True;
```

Remarks

When set to True, the macro will be expanded, otherwise macro definition is replaced by null string. You can use the Active property to modify the SQL property.

The default value is True.

Example

```
IBCQuery.SQL.Text := 'SELECT * FROM Dept WHERE DeptNo > 20 &Cond1';  
IBCQuery.Macros[0].Value := 'and DName is NULL';  
IBCQuery.Macros[0].Active:= False;
```

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.20.2.2 AsDateTime Property

Used to set the TDateTime value to a macro.

Class

[TMacro](#)

Syntax

```
property AsDateTime: TDateTime;
```

Remarks

Use the AsDateTime property to set the TDateTime value to a macro.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.20.2.3 AsFloat Property

Used to set the float value to a macro.

Class

[TMacro](#)

Syntax

```
property AsFloat: double;
```

Remarks

Use the AsFloat property to set the float value to a macro.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.20.2.4 AsInteger Property

Used to set the integer value to a macro.

Class

[TMacro](#)

Syntax

```
property AsInteger: integer;
```

Remarks

Use the AsInteger property to set the integer value to a macro.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.20.2.5 AsString Property

Used to assign the string value to a macro.

Class

[TMacro](#)

Syntax

```
property AsString: string;
```

Remarks

Use the AsString property to assign the string value to a macro. Read the AsString property to determine the value of macro represented as a string.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.20.2.6 Name Property

Used to identify a particular macro.

Class

[TMacro](#)

Syntax

```
property Name: string;
```

Remarks

Use the Name property to identify a particular macro.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.20.2.7 Value Property

Used to set the value to a macro.

Class

[TMacro](#)

Syntax

```
property value: string;
```

Remarks

Use the Value property to set the value to a macro.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.21 TMacros Class

Controls a list of TMacro objects for the [TCustomDASQL.Macros](#) or [TCustomDADataset](#) components.

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

Unit

[DBAccess](#)

Syntax

```
TMacros = class(TCollection);
```

Remarks

Use TMacros to manage a list of TMacro objects for the [TCustomDASQL](#) or [TCustomDADataset](#) components.

See Also

- [TMacro](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.21.1 Members

[TMacros](#) class overview.

Properties

Name	Description
Items	Used to iterate through all the macros parameters.

Methods

Name	Description
AssignValues	Copies the macros values and properties from the specified source.
Expand	Changes the macros in the passed SQL statement to their values.
FindMacro	Finds a macro with the specified name.
IsEqual	Compares itself with another TMacro object.
MacroByName	Used to search for a macro with the specified name.
Scan	Creates a macros from the passed SQL statement.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.21.2 Properties

Properties of the **TMacros** class.

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

Public

Name	Description
Items	Used to iterate through all the macros parameters.

See Also

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

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.21.2.1 Items Property(Indexer)

Used to iterate through all the macros parameters.

Class

[TMacros](#)

Syntax

```
property Items[Index: integer]: TMacro; default;
```

Parameters

Index

Holds the index in the range 0..Count - 1.

Remarks

Use the Items property to iterate through all macros parameters. Index identifies the index in the range 0..Count - 1.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.21.3 Methods

Methods of the **TMacros** class.

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

Public

Name	Description
AssignValues	Copies the macros values and properties from the specified source.
Expand	Changes the macros in the passed SQL statement to their values.
FindMacro	Finds a macro with the specified name.
IsEqual	Compares itself with another TMacro object.
MacroByName	Used to search for a macro

	with the specified name.
Scan	Creates a macros from the passed SQL statement.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.21.3.1 AssignValues Method

Copies the macros values and properties from the specified source.

Class

[TMacros](#)

Syntax

```
procedure AssignValues(Value: TMacros);
```

Parameters

Value

Holds the source to copy the macros values and properties from.

Remarks

The Assign method copies the macros values and properties from the specified source. Macros are not recreated. Only the values of macros with matching names are assigned.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.21.3.2 Expand Method

Changes the macros in the passed SQL statement to their values.

Class

[TMacros](#)

Syntax

```
procedure Expand(var SQL: string);
```

Parameters

SQL

Holds the passed SQL statement.

Remarks

Call the Expand method to change the macros in the passed SQL statement to their values.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.21.3.3 FindMacro Method

Finds a macro with the specified 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 is found, nil otherwise.

Remarks

Call the FindMacro method to find a macro with the specified name. If a match is found, FindMacro returns the macro. Otherwise, it returns nil. Use this method instead of a direct reference to the [Items](#) property to avoid depending on the order of the items.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.21.3.4 IsEqual Method

Compares itself with another TMacro object.

Class

[TMacros](#)

Syntax

```
function IsEqual(value: TMacros): boolean;
```

Parameters

Value

Holds the values of TMacro objects.

Return Value

True, if the number of TMacro objects and the values of all TMacro objects are equal.

Remarks

Call the IsEqual method to compare itself with another TMacro object. Returns True if the number of TMacro objects and the values of all TMacro objects are equal.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.21.3.5 MacroByName Method

Used to search for a macro with the specified name.

Class

[TMacros](#)

Syntax

```
function MacroByName(const value: string): TMacro;
```

Parameters

Value

Holds a name of the macro to search for.

Return Value

TMacro object, if a macro with specified name was found.

Remarks

Call the `MacroByName` method to find a `Macro` with the name passed in `Value`. If a match is found, `MacroByName` returns the `Macro`. Otherwise, an exception is raised. Use this method instead of a direct reference to the [Items](#) property to avoid depending on the order of the items.

To locate a macro by name without raising an exception if the parameter is not found, use the [FindMacro](#) method.

To set a value to a macro, use the [TMacro.Value](#) property.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.21.3.6 Scan Method

Creates a macros from the passed SQL statement.

Class

[TMacros](#)

Syntax

```
procedure Scan(const 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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.22 TPoolingOptions Class

This class allows setting up the behaviour of the connection pool.

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

Unit

[DBAccess](#)

Syntax

```
TPoolingOptions = class(TPersistent);
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.22.1 Members

[TPoolingOptions](#) class overview.

Properties

Name	Description
ConnectionLifetime	Used to specify the maximum time during which an open connection can be used by connection pool.
MaxPoolSize	Used to specify the maximum number of connections that can be opened in connection pool.
MinPoolSize	Used to specify the minimum number of connections that can be opened in the connection pool.
PoolId	Used to specify an ID for a connection pool.
Validate	Used for a connection to be validated when it is returned from the pool.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.22.2 Properties

Properties of the **TPoolingOptions** class.

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

Published

Name	Description
ConnectionLifetime	Used to specify the maximum time during which an open connection can be used by connection pool.
MaxPoolSize	Used to specify the maximum number of connections that can be opened in connection pool.
MinPoolSize	Used to specify the minimum number of connections that can be opened in the connection pool.
PoolId	Used to specify an ID for a connection pool.
Validate	Used for a connection to be validated when it is returned from the pool.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.22.2.1 ConnectionLifetime Property

Used to specify the maximum time during which an open connection can be used by connection pool.

Class

[TPoolingOptions](#)

Syntax

```
property ConnectionLifetime: integer default  
DefValConnectionLifetime;
```

Remarks

Use the ConnectionLifeTime property to specify the maximum time during which an open connection can be used by connection pool. Measured in milliseconds. Pool deletes connections with exceeded connection lifetime when [TCustomDACConnection](#) is about to close. If ConnectionLifetime is set to 0 (by default), then the lifetime of connection is infinite. ConnectionLifetime concerns only inactive connections in the pool.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.22.2.2 MaxPoolSize Property

Used to specify the maximum number of connections that can be opened in connection pool.

Class

[TPoolingOptions](#)

Syntax

```
property MaxPoolSize: integer default DefValMaxPoolSize;
```

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.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.22.2.3 MinPoolSize Property

Used to specify the minimum number of connections that can be opened in the connection pool.

Class

[TPoolingOptions](#)

Syntax

```
property MinPoolSize: integer default DefValMinPoolSize;
```

Remarks

Use the MinPoolSize property to specify the minimum number of connections that can be opened in the connection pool.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.22.2.4 PoolId Property

Used to specify an ID for a connection pool.

Class

[TPoolingOptions](#)

Syntax

```
property PoolId: Integer default DefValPoolId;
```

Remarks

Use the PoolId property to make a group of connections use a specific connection pool.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.22.2.5 Validate Property

Used for a connection to be validated when it is returned from the pool.

Class

[TPoolingOptions](#)

Syntax

```
property validate: boolean default DefValValidate;
```

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.23 TSmartFetchOptions Class

Smart fetch options are used to set up the behavior of the SmartFetch mode.

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

Unit

[DBAccess](#)

Syntax

```
TSmartFetchOptions = class(TPersistent);
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.23.1 Members

[TSmartFetchOptions](#) class overview.

Properties

Name	Description
Enabled	Sets SmartFetch mode enabled or not.
LiveBlock	Used to minimize memory consumption.
PrefetchedFields	List of fields additional to key fields that will be read from the database on dataset open.

SQLGetKeyValues	SQL query for the read key and prefetched fields from the database.
---------------------------------	---

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.23.2 Properties

Properties of the **TSmartFetchOptions** class.

For a complete list of the **TSmartFetchOptions** class members, see the

[TSmartFetchOptions Members](#) topic.

Published

Name	Description
Enabled	Sets SmartFetch mode enabled or not.
LiveBlock	Used to minimize memory consumption.
PrefetchedFields	List of fields additional to key fields that will be read from the database on dataset open.
SQLGetKeyValues	SQL query for the read key and prefetched fields from the database.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.23.2.1 Enabled Property

Sets SmartFetch mode enabled or not.

Class

[TSmartFetchOptions](#)

Syntax

```
property Enabled: Boolean default False;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.23.2.2 LiveBlock Property

Used to minimize memory consumption.

Class

[TSmartFetchOptions](#)

Syntax

```
property LiveBlock: Boolean default True;
```

Remarks

If LiveBlock is True, then on navigating through a dataset forward or backward, memory will be allocated for records count defined in the the FetchRows property, and no additional memory will be allocated. But if you return records that were read from the database before, they will be read from the database again, because when you left block with these records, memory was free. So the LiveBlock mode minimizes memory consumption, but can decrease performance, because it can lead to repeated data reading from the database.

The default value of LiveBlock is False.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.23.2.3 PrefetchedFields Property

List of fields additional to key fields that will be read from the database on dataset open.

Class

[TSmartFetchOptions](#)

Syntax

```
property PrefetchedFields: string;
```

Remarks

If you are going to use locate, filter or sort by some fields, then these fields should be added to the prefetched fields list to avoid excessive reading from the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.1.23.2.4 SQLGetKeyValues Property

SQL query for the read key and prefetched fields from the database.

Class

[TSmartFetchOptions](#)

Syntax

```
property SQLGetKeyValues: TStrings;
```

Remarks

SQLGetKeyValues is used when the basic SQL query is complex and the query for reading the key and prefetched fields can't be generated automatically.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.2 Types

Types in the **DBAccess** unit.

Types

Name	Description
TAfterExecuteEvent	This type is used for the TCustomDADataset.AfterExecute and TCustomDASQL.AfterExecute

	te events.
TAfterFetchEvent	This type is used for the TCustomDADataset.AfterFetch event.
TBeforeFetchEvent	This type is used for the TCustomDADataset.BeforeFetch event.
TConnectionLostEvent	This type is used for the TCustomDAConnection.OnConnectionLost event.
TDAConnectionErrorEvent	This type is used for the TCustomDAConnection.OnError event.
TDATransactionErrorEvent	This type is used for the TDATransaction.OnError event.
TRefreshOptions	Represents the set of TRefreshOption .
TUpdateExecuteEvent	This type is used for the TCustomDADataset.AfterUpdateExecute and TCustomDADataset.BeforeUpdateExecute events.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.2.1 TAfterExecuteEvent Procedure Reference

This type is used for the [TCustomDADataset.AfterExecute](#) and [TCustomDASQL.AfterExecute](#) events.

Unit

[DBAccess](#)

Syntax

```
TAfterExecuteEvent = procedure (Sender: TObject; Result: boolean)
of object;
```

Parameters

Sender

An object that raised the event.

Result

The result is True if SQL statement is executed successfully. False otherwise.

© 1997-2024

Devart. All Rights

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.10.2.2 TAfterFetchEvent Procedure Reference

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

Unit

[DBAccess](#)

Syntax

```
TAfterFetchEvent = procedure (DataSet: TCustomDADataset) of  
object;
```

Parameters

DataSet

Holds the TCustomDADataset descendant to synchronize the record position with.

© 1997-2024

Devart. All Rights

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.10.2.3 TBeforeFetchEvent Procedure Reference

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

Unit

[DBAccess](#)

Syntax

```
TBeforeFetchEvent = procedure (DataSet: TCustomDADataset; var  
Cancel: boolean) of object;
```

Parameters

DataSet

Holds the TCustomDADataset descendant to synchronize the record position with.

Cancel

True, if the current fetch operation should be aborted.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.2.4 TConnectionLostEvent Procedure Reference

This type is used for the [TCustomDACConnection.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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.2.5 TDAConnectionErrorEvent Procedure Reference

This type is used for the [TCustomDACConnection.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 .

© 1997-2024

Devart. All Rights

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.10.2.6 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.

© 1997-2024

Devart. All Rights

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.10.2.7 TRefreshOptions Set

Represents the set of [TRefreshOption](#).

Unit

[DBAccess](#)

Syntax

```
TRefreshOptions = set of TRefreshOption;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.2.8 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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.3 Enumerations

Enumerations in the **DBAccess** unit.

Enumerations

Name	Description
TLabelSet	Sets the language of labels in the connect dialog.
TLockMode	Specifies the lock mode.

TRefreshOption	Indicates when the editing record will be refreshed.
TRetryMode	Specifies the application behavior when connection is lost.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.3.1 TLabelSet Enumeration

Sets the language of labels in the connect dialog.

Unit

[DBAccess](#)

Syntax

```
TLabelSet = (IsCustom, IsEnglish, IsFrench, IsGerman, IsItalian, IsPolish, IsPortuguese, IsRussian, IsSpanish);
```

Values

Value	Meaning
IsCustom	Set the language of labels in the connect dialog manually.
IsEnglish	Set English as the language of labels in the connect dialog.
IsFrench	Set French as the language of labels in the connect dialog.
IsGerman	Set German as the language of labels in the connect dialog.
IsItalian	Set Italian as the language of labels in the connect dialog.
IsPolish	Set Polish as the language of labels in the connect dialog.
IsPortuguese	Set Portuguese as the language of labels in the connect dialog.
IsRussian	Set Russian as the language of labels in the connect dialog.
IsSpanish	Set Spanish as the language of labels in the connect dialog.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.3.2 TLockMode Enumeration

Specifies the lock mode.

Unit

[DBAccess](#)

Syntax

```
TLockMode = (TmNone);
```

Values

Value	Meaning
ImLockDelayed	Locking occurs when the user posts an edited record, then the lock is released. Locking is done by the RefreshRecord method. Corresponds to optimistic locking.
ImLockImmediate	Locking occurs when the user starts editing a record. The lock is released after the user has posted or canceled the changes. Corresponds to pessimistic locking.
ImNone	No locking occurs. This mode should only be used in single user applications. The default value.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.3.3 TRefreshOption Enumeration

Indicates when the editing record will be refreshed.

Unit

[DBAccess](#)

Syntax

```
TRefreshOption = (roAfterInsert, roAfterUpdate, roBeforeEdit);
```

Values

Value	Meaning
roAfterInsert	Refresh is performed after inserting.
roAfterUpdate	Refresh is performed after updating.

roBeforeEdit	Refresh is performed by Edit method.
---------------------	--------------------------------------

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.3.4 TRetryMode Enumeration

Specifies the application behavior when connection is lost.

Unit

[DBAccess](#)

Syntax

```
TRetryMode = (rmRaise, rmReconnect, rmReconnectExecute);
```

Values

Value	Meaning
rmRaise	An exception is raised.
rmReconnect	Reconnect is performed and then exception is raised.
rmReconnectExecute	Reconnect is performed and abortive operation is reexecuted. Exception is not raised.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.4 Variables

Variables in the **DBAccess** unit.

Variables

Name	Description
ChangeCursor	When set to True allows data access components to change screen cursor for the execution time.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.10.4.1 ChangeCursor Variable

When set to True allows data access components to change screen cursor for the execution time.

Unit

[DBAccess](#)

Syntax

```
changeCursor: boolean = True;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11 IBC

This unit contains main components of IBDAC.

Classes

Name	Description
TCustomIBCDataset	A base class that defines InterBase functionality for a dataset.
TCustomIBCQuery	A base class for defining functionality for descendant classes which access database using SQL statements.
TCustomIBCTable	A base class that defines functionality for descendant classes which access data in a single table without writing SQL statements.
TIBCArray	A class representing the value of the InterBase array data type.
TIBCArrayField	A class encapsulating the fundamental behavior common to the InterBase array fields.

TIBConnection	A component for setting and controlling connections to an InterBase database.
TIBConnectionOptions	This class allows setting up the behaviour of the TIBConnection class.
TIBDataSetOptions	This class allows setting up the behaviour of the TIBDataSet class.
TIBDataSource	TIBDataSource provides an interface between an IBDAC dataset components and data-aware controls on a form.
TIBDbKeyField	A class representing the InterBase RDB\$DB_KEY field.
TIBEncryptor	The class that performs encrypting and decrypting of data.
TIBMetaData	A component for obtaining metainformation about database objects from the server.
TIBParam	A class that is used to set the values of individual parameters passed with queries or stored procedures.
TIBParams	Used to control TIBParam objects.
TIBQuery	A component for executing queries and operating record sets. It also provides flexible way to update data.
TIBSQL	A component for executing SQL statements and calling stored procedures on the database server.
TIBSSLConnectionOptions	A class for setting up the SSL options.
TIBStoredProc	A component for accessing and executing stored procedures and functions.

TIBCTable	A component for retrieving and updating data in a single table without writing SQL statements.
TIBCTransaction	A component for managing transactions in an application.
TIBCUUpdateSQL	A component for tuning update operations for the DataSet component.

Types

Name	Description
TIBCTransactionErrorEvent	This type is used for the TIBCTransaction.OnError event.

Enumerations

Name	Description
TGeneratorMode	Specifies the method used internally to generate a sequenced field.
TIBCProtocol	Specifies the network protocol of connection with InterBase server.
TIBCTransactionAction	Specifies the transaction behaviour when connection closes.

Variables

Name	Description
Connections	Holds pointers to all TIBConnection objects of an application.
DefConnection	Read this variable to get pointer to default connection object. Same as DefaultConnection function.
UseDefConnection	When set to true enables

	TCustomIBCDataSet and TIBCSQL components to use default connection if they are not attached to any connection.
--	--

Constants

Name	Description
IBDACVersion	Read this constant to get current version number for IBDAC.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1 Classes

Classes in the **IBC** unit.

Classes

Name	Description
TCustomIBCDataSet	A base class that defines InterBase functionality for a dataset.
TCustomIBCQuery	A base class for defining functionality for descendant classes which access database using SQL statements.
TCustomIBCTable	A base class that defines functionality for descendant classes which access data in a single table without writing SQL statements.
TIBCArray	A class representing the value of the InterBase array data type.
TIBCArrayField	A class encapsulating the fundamental behavior common to the InterBase array fields.

TIBCConnection	A component for setting and controlling connections to an InterBase database.
TIBCConnectionOptions	This class allows setting up the behaviour of the TIBCConnection class.
TIBCDatasetOptions	This class allows setting up the behaviour of the TIBCDataset class.
TIBCDatasource	TIBCDatasource provides an interface between an IBDAC dataset components and data-aware controls on a form.
TIBCDbKeyField	A class representing the InterBase RDB\$DB_KEY field.
TIBCEncryptor	The class that performs encrypting and decrypting of data.
TIBCMetaData	A component for obtaining metainformation about database objects from the server.
TIBCPParam	A class that is used to set the values of individual parameters passed with queries or stored procedures.
TIBCPParams	Used to control TIBCPParam objects.
TIBCQuery	A component for executing queries and operating record sets. It also provides flexible way to update data.
TIBCSQL	A component for executing SQL statements and calling stored procedures on the database server.
TIBCSSLConnectionOptions	A class for setting up the SSL options.
TIBCStoredProc	A component for accessing and executing stored procedures and functions.

TIBCTable	A component for retrieving and updating data in a single table without writing SQL statements.
TIBCTransaction	A component for managing transactions in an application.
TIBCUpdateSQL	A component for tuning update operations for the DataSet component.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1 TCustomIBCDataSet Class

A base class that defines InterBase functionality for a dataset.

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

Unit

[IBC](#)

Syntax

```
TCustomIBCDataSet = class (TCustomDADataset);
```

Remarks

TCustomIBCDataSet is a component that defines InterBase functionality for a dataset.

TCustomIBCDataSet can execute queries, fetch rows and controls InterBase specific data types. Applications never use TCustomIBCDataSet objects directly. Instead they use the descendants of TCustomIBCDataSet, such as TIBCQuery, TIBCStoredProc and TIBCTable, which inherit its database-related properties and methods.

TIBCQuery provides insert, delete and update operations on records by dynamically generated SQL statements. It uses [TCustomDADataset.KeyFields](#) property to build SQL statements for the SQLDelete, SQLInsert and SQLUpdate properties if they were empty before updating the database.

Inheritance Hierarchy

[TMemDataSet](#)[TCustomDADataset](#)**TCustomIBCDataset**

See Also

- [TCustomDADataset.KeyFields](#)
- [TIBCQuery](#)
- [TIBCStoredProc](#)
- [TIBCTable](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.11.1.1.1 Members

[TCustomIBCDataset](#) class overview.

Properties

Name	Description
AutoCommit	Used to automatically commit each update, insert or delete statement by database server.
BaseSQL (inherited from TCustomDADataset)	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions (inherited from TCustomDADataset)	Used to add WHERE conditions to a query
Connection	Used to specify the connection in which the dataset will be executed.
Cursor	Used for positioned UPDATE and DELETE statements made for the data retrieved with the

	SELECT statements with the FOR UPDATE clause.
DataTypeMap (inherited from TCustomDADataset)	Used to set data type mapping rules
Debug (inherited from TCustomDADataset)	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields (inherited from TCustomDADataset)	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
Disconnected (inherited from TCustomDADataset)	Used to keep dataset opened after connection is closed.
DMLRefresh	Used to refresh record by the RETURNING clause when insert is performed.
Encryption	Used to specify encryption options in a dataset.
ExplainPlan	Used to obtain the query execution plan for Table, Query, and SQL components.
FetchAll	Used to retrieve all records in a dataset.
FetchRows (inherited from TCustomDADataset)	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL (inherited from TCustomDADataset)	Used to change the WHERE clause of SELECT statement and reopen a query.
FinalSQL (inherited from TCustomDADataset)	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
GeneratorMode	Used to specify which method is used internally to generate a sequenced field.
GeneratorStep	Used to set the increment for

	increasing or decreasing current generator value when using the automatic key field value generation feature.
Handle	Used to specify the handle for the SQL statement of TCustomIBCDataset.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
IsQuery	Used to check if the SQL statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
KeyFields (inherited from TCustomDADataset)	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
KeyGenerator	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
LockMode	Used to indicate when to perform a locking of editing record.
MacroCount (inherited from TCustomDADataset)	Used to get the number of macros associated with the Macros property.
Macros (inherited from TCustomDADataset)	Makes it possible to change SQL queries easily.
MasterFields (inherited from TCustomDADataset)	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)	Used to specify the data source component which binds current dataset to the master one.
Options	Used to specify the behaviour of the TCustomIBCDataSet object.
ParamCheck (inherited from TCustomDADataset)	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
ParamCount (inherited from TCustomDADataset)	Used to indicate how many parameters are there in the Params property.
Params (inherited from TCustomDADataset)	Used to view and set parameter names, values, and data types dynamically.
Plan	Used to get or set the PLAN clause of the SELECT statement.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
ReadOnly (inherited from TCustomDADataset)	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions (inherited from TCustomDADataset)	Used to indicate when the editing record is refreshed.
RowsAffected (inherited from TCustomDADataset)	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
RowsDeleted	Used to indicate the number of rows that were deleted during the last query operation.

RowsFetched	Used to get the number of the currently fetched rows.
RowsInserted	Used to indicate the number of rows that were inserted during the last query operation.
RowsUpdated	Used to indicate the number of rows that were updated during the last query operation.
SmartFetch	The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.
SQL (inherited from TCustomDADataset)	Used to provide a SQL statement that a query component executes when its Open method is called.
SQLDelete (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying a deletion to a record.
SQLInsert (inherited from TCustomDADataset)	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
SQLLock (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to perform a record lock.
SQLRecCount (inherited from TCustomDADataset)	Used to specify the SQL statement that is used to get the record count when opening a dataset.
SQLRefresh (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.
SQLType	Used to get the typecode of the SQL statement being processed by the InterBase database server.

SQLUpdate (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying an update to a dataset.
Transaction	Used to determine the transaction under which the query of this dataset executes.
UniDirectional (inherited from TCustomDADataset)	Used if an application does not need bidirectional access to records in the result set.
UpdateObject	Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.
UpdateRecordTypes (inherited from TMemDataSet)	Used to indicate the update status for the current record when cached updates are enabled.
UpdatesPending (inherited from TMemDataSet)	Used to check the status of the cached updates buffer.
UpdateTransaction	Used to get or set the transaction for modifying a dataset.

Methods

Name	Description
AddWhere (inherited from TCustomDADataset)	Adds condition to the WHERE clause of SELECT statement in the SQL property.
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
BreakExec (inherited from TCustomDADataset)	Breaks execution of the SQL statement on the server.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.

CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
CreateBlobStream (inherited from TCustomDADataset)	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
CreateProcCall	Assigns PL/SQL block that calls stored procedure to the SQL property.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
DeleteWhere (inherited from TCustomDADataset)	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
Execute (inherited from TCustomDADataset)	Overloaded. Executes a SQL statement on the server.
Executing (inherited from TCustomDADataset)	Indicates whether SQL statement is still being executed.
Fetched (inherited from TCustomDADataset)	Used to find out whether TCustomDADataset has fetched all rows.
Fetching (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is still fetching rows.
FetchingAll (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is fetching all rows to the end.
FindKey (inherited from TCustomDADataset)	Searches for a record which contains specified field values.

FindMacro (inherited from TCustomDADataset)	Finds a macro with the specified name.
FindNearest (inherited from TCustomDADataset)	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.
FindParam	Determines if a parameter with the specified name exists in a dataset.
GetArray	Retrieves a TIBCArry object for a field when only its name is known.
GetBlob	Retrieves a TIBCBlob object for a field when only its name is known.
GetDataType (inherited from TCustomDADataset)	Returns internal field types defined in the MemData and accompanying modules.
GetFieldObject (inherited from TCustomDADataset)	Returns a multireference shared object from field.
GetFieldPrecision (inherited from TCustomDADataset)	Retrieves the precision of a number field.
GetFieldScale (inherited from TCustomDADataset)	Retrieves the scale of a number field.
GetKeyFieldNames (inherited from TCustomDADataset)	Provides a list of available key field names.
GetOrderBy (inherited from TCustomDADataset)	Retrieves an ORDER BY clause from a SQL statement.
GotoCurrent (inherited from TCustomDADataset)	Sets the current record in this dataset similar to the current record in another dataset.
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.
Lock (inherited from TCustomDADataset)	Locks the current record.
MacroByName (inherited from TCustomDADataset)	Finds a macro with the specified name.
ParamByName	Called to set or use parameter information for a specific parameter based on its name.
Prepare (inherited from TCustomDADataset)	Allocates, opens, and parses cursor for a query.
RefreshRecord (inherited from TCustomDADataset)	Actualizes field values for the current record.
RestoreSQL (inherited from TCustomDADataset)	Restores the SQL property modified by AddWhere and SetOrderBy.
RestoreUpdates (inherited from TMemDataSet)	Marks all records in the cache of updates as unapplied.
RevertRecord (inherited from TMemDataSet)	Cancel changes made to the current record when cached updates are enabled.
SaveSQL (inherited from TCustomDADataset)	Saves the SQL property value to BaseSQL.
SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
SetOrderBy (inherited from TCustomDADataset)	Builds an ORDER BY clause of a SELECT statement.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.

SQLSaved (inherited from TCustomDADataset)	Determines if the SQL property value was saved to the BaseSQL property.
UnLock (inherited from TCustomDADataset)	Releases a record lock.
UnPrepare (inherited from TMemDataSet)	Frees the resources allocated for a previously prepared query on the server and client sides.
UpdateResult (inherited from TMemDataSet)	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
UpdateStatus (inherited from TMemDataSet)	Indicates the current update status for the dataset when cached updates are enabled.

Events

Name	Description
AfterExecute (inherited from TCustomDADataset)	Occurs after a component has executed a query to database.
AfterFetch (inherited from TCustomDADataset)	Occurs after dataset finishes fetching data from server.
AfterUpdateExecute (inherited from TCustomDADataset)	Occurs after executing insert, delete, update, lock and refresh operations.
BeforeFetch (inherited from TCustomDADataset)	Occurs before dataset is going to fetch block of records from the server.
BeforeUpdateExecute (inherited from TCustomDADataset)	Occurs before executing insert, delete, update, lock, and refresh operations.
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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.11.1.1.2 Properties

Properties of the **TCustomIBDataSet** class.

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

Public

Name	Description
AutoCommit	Used to automatically commit each update, insert or delete statement by database server.
BaseSQL (inherited from TCustomDADataset)	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions (inherited from TCustomDADataset)	Used to add WHERE conditions to a query
Connection	Used to specify the connection in which the dataset will be executed.
Cursor	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
DataTypeMap (inherited from TCustomDADataset)	Used to set data type mapping rules
Debug (inherited from TCustomDADataset)	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields (inherited from TCustomDADataset)	Used to specify the fields that correspond to the foreign key fields from

	MasterFields when building master/detail relationship.
Disconnected (inherited from TCustomDADataset)	Used to keep dataset opened after connection is closed.
DMLRefresh	Used to refresh record by the RETURNING clause when insert is performed.
Encryption	Used to specify encryption options in a dataset.
ExplainPlan	Used to obtain the query execution plan for Table, Query, and SQL components.
FetchAll	Used to retrieve all records in a dataset.
FetchRows (inherited from TCustomDADataset)	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL (inherited from TCustomDADataset)	Used to change the WHERE clause of SELECT statement and reopen a query.
FinalSQL (inherited from TCustomDADataset)	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
GeneratorMode	Used to specify which method is used internally to generate a sequenced field.
GeneratorStep	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
Handle	Used to specify the handle for the SQL statement of TCustomIBCDataSet.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
IsQuery	Used to check if the SQL

	statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
KeyFields (inherited from TCustomDADataset)	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
KeyGenerator	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
LockMode	Used to indicate when to perform a locking of editing record.
MacroCount (inherited from TCustomDADataset)	Used to get the number of macros associated with the Macros property.
Macros (inherited from TCustomDADataset)	Makes it possible to change SQL queries easily.
MasterFields (inherited from TCustomDADataset)	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)	Used to specify the data source component which binds current dataset to the master one.
Options	Used to specify the behaviour of the TCustomIBCDataset object.

ParamCheck (inherited from TCustomDADataset)	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
ParamCount (inherited from TCustomDADataset)	Used to indicate how many parameters are there in the Params property.
Params (inherited from TCustomDADataset)	Used to view and set parameter names, values, and data types dynamically.
Plan	Used to get or set the PLAN clause of the SELECT statement.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
ReadOnly (inherited from TCustomDADataset)	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions (inherited from TCustomDADataset)	Used to indicate when the editing record is refreshed.
RowsAffected (inherited from TCustomDADataset)	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
RowsDeleted	Used to indicate the number of rows that were deleted during the last query operation.
RowsFetched	Used to get the number of the currently fetched rows.
RowsInserted	Used to indicate the number of rows that were inserted during the last query operation.
RowsUpdated	Used to indicate the number of rows that were updated during the last query operation.
SmartFetch	The SmartFetch mode is used for fast navigation

	through a huge amount of records and to minimize memory consumption.
SQL (inherited from TCustomDADataset)	Used to provide a SQL statement that a query component executes when its Open method is called.
SQLDelete (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying a deletion to a record.
SQLInsert (inherited from TCustomDADataset)	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
SQLLock (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to perform a record lock.
SQLRecCount (inherited from TCustomDADataset)	Used to specify the SQL statement that is used to get the record count when opening a dataset.
SQLRefresh (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.
SQLType	Used to get the typecode of the SQL statement being processed by the InterBase database server.
SQLUpdate (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying an update to a dataset.
Transaction	Used to determine the transaction under which the query of this dataset executes.
UniDirectional (inherited from TCustomDADataset)	Used if an application does not need bidirectional access to records in the result set.

UpdateObject	Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.
UpdateRecordTypes (inherited from TMemDataSet)	Used to indicate the update status for the current record when cached updates are enabled.
UpdatesPending (inherited from TMemDataSet)	Used to check the status of the cached updates buffer.
UpdateTransaction	Used to get or set the transaction for modifying a dataset.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.1 AutoCommit Property

Used to automatically commit each update, insert or delete statement by database server.

Class

[TCustomIBCDataSet](#)

Syntax

```
property AutoCommit: boolean;
```

Remarks

When True and Connection.AutoCommit is True, each update, insert or delete statement is automatically committed by database server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.2 Connection Property

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

Class

[TCustomIBCDataset](#)

Syntax

```
property Connection: TIBConnection;
```

Remarks

Use the Connection property to specify the connection in which the dataset will be executed. If connection is not connected, the Open method calls Connection.Connect.

See Also

- [TIBConnection](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.3 Cursor Property

Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.

Class

[TCustomIBCDataset](#)

Syntax

```
property Cursor: string;
```

Remarks

Use the Cursor property for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause. FetchRows property must be set to 1 for using cursor.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.11.1.1.2.4 DMLRefresh Property

Used to refresh record by the RETURNING clause when insert is performed.

Class

[TCustomIBCDataset](#)

Syntax

```
property DMLRefresh: boolean;
```

Remarks

Use the DMLRefresh property to refresh record by the RETURNING clause when insert is performed. This feature is only for Firebird 2.0 and higher.

The default value is False.

Note: When the DMLRefresh property is set to True, the value of

[TCustomDADataset.RefreshOptions](#) is ignored to avoid refetching field values from the server.

See Also

- [Updating Data with IBDAC Dataset Components](#)
- [TCustomDADataset.RefreshOptions](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.5 Encryption Property

Used to specify encryption options in a dataset.

Class

[TCustomIBCDataset](#)

Syntax

```
property Encryption: TIBCEncryption;
```

Remarks

Set the Encryption options for using encryption in a dataset.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.6 ExplainPlan Property

Used to obtain the query execution plan for Table, Query, and SQL components.

Class

[TCustomIBCDataset](#)

Syntax

```
property ExplainPlan: string;
```

Remarks

Use the ExplainPlan property to obtain the query execution plan for Table, Query, and SQL components.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.7 FetchAll Property

Used to retrieve all records in a dataset.

Class

[TCustomIBCDataset](#)

Syntax

```
property FetchAll: boolean;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.8 GeneratorMode Property

Used to specify which method is used internally to generate a sequenced field.

Class

[TCustomIBCDataset](#)

Syntax

```
property GeneratorMode: TGeneratorMode default gmPost;
```

Remarks

Set the GeneratorMode property to specify which method is used internally to generate a sequenced field.

See Also

- [KeyGenerator](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.9 GeneratorStep Property

Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.

Class

[TCustomIBCDataset](#)

Syntax

```
property GeneratorStep: integer default 1;
```

Remarks

Use the GeneratorStep property to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature. The default value is 1.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.11.1.1.2.10 Handle Property

Used to specify the handle for the SQL statement of TCustomIBCDataset.

Class

[TCustomIBCDataset](#)

Syntax

```
property Handle: TISC_STMT_HANDLE;
```

Remarks

Use the Handle property to specify the handle for the SQL statement of TCustomIBCDataset.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.11 IsQuery Property

Used to check if the SQL statement returns rows.

Class

[TCustomIBCDataset](#)

Syntax

```
property IsQuery: boolean;
```

Remarks

When the TCustomIBCDataset component is prepared, returns True, if the SQL statement is a SELECT block that returns the REF CURSOR parameter.

Use the IsQuery property to check whether the SQL statement returns rows or not.

TCustomIBCDataset returns rows when the SQL statement is the SELECT or PL/SQL block with the REF CURSOR parameter. TCustomIBCDataset must be prepared before.

IsQuery is a read-only property.

© 1997-2024

Devart. All Rights

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.11.1.1.2.12 KeyGenerator Property

Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.

Class

[TCustomIBCDataSet](#)

Syntax

```
property KeyGenerator: string;
```

Remarks

Use the KeyGenerator property to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.

Note: KeyGenerator is used by TCustomIBCDataSet only if [TCustomDADDataSet.KeyFields](#) property is assigned.

See Also

- [GeneratorMode](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.13 LockMode Property

Used to indicate when to perform a locking of editing record.

Class

[TCustomIBCDataSet](#)

Syntax

```
property LockMode: TLockMode default 1mNone;
```

Remarks

Use the LockMode property to define when to perform locking of an editing record. Locking a

record is useful in creating multi-user applications. It prevents modification of a record by several users at the same time. Locking realizes through execution of the SELECT FOR UPDATE statement in Firebird 1.5 and through the UPDATE operation in other database servers.

See Also

- [TCustomDADDataSet.Lock](#)
- [TCustomDADDataSet.SQLLock](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.14 Options Property

Used to specify the behaviour of the TCustomIBCDataSet object.

Class

[TCustomIBCDataSet](#)

Syntax

```
property Options: TIBCDataSetOptions;
```

Remarks

Set the properties of Options to specify behaviour of a TCustomIBCDataSet object.

Descriptions of all options are in the table below.

Option Name	Description
AutoClose	Used to for CustomIBCDataSet to close cursor after fetching all rows.
BooleanDomainFields	Used to create TBooleanField for fields that have domain of the integer data type, and the domain name contains 'BOOLEAN'.
CacheArrays	Used to allocate local memory buffer for a copy of the array.
CacheBlobs	Used to allocate local memory buffer for a copy of the BLOB.
ComplexArrayFields	Used to store array fields as

	TIBCArrayField objects.
DefaultValues	Used for TCustomIBCDataSet to fill the DefaultExpression property of TField objects by the appropriate value.
DeferredArrayRead	Used for fetching all InterBase array values when they are explicitly requested.
DeferredBlobRead	Used for fetching all InterBase BLOB values when they are explicitly requested.
DescribeParams	Used to specify whether to query TIBCPParam properties from the server when executing the TCustomDADataset.Prepare method.
ExtendedFieldsInfo	Used to perform an additional query to get information about the returned fields and the tables they belong to.
FieldsAsString	Used to treat all non-BLOB fields as being of string datatype.
FullRefresh	Used to refresh fields from all tables of the query.
PrepareUpdateSQL	Used to automatically prepare update queries before execution.
QueryRowsAffected	Used to increase the performance of update operations.
SetDomainNames	Used to retrieve the DOMAIN name for a field.
SetEmptyStrToNull	Force replace of empty strings with NULL values in data. The default value is False.
StreamedBlobs	Used to handle and save BLOBs as streamed BLOBs.
StrictUpdate	Used for TCustomIBCDataSet to raise an exception when the number of the updated or deleted records is not equal 1.

See Also

- [TCustomDADataset.Options](#)
- [BLOB Data Types](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.15 Plan Property

Used to get or set the PLAN clause of the SELECT statement.

Class

[TCustomIBCDataset](#)

Syntax

```
property Plan: string;
```

Remarks

Use the Plan property to get or set the PLAN clause of the SELECT statement.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.16 RowsDeleted Property

Used to indicate the number of rows that were deleted during the last query operation.

Class

[TCustomIBCDataset](#)

Syntax

```
property RowsDeleted: integer;
```

Remarks

Check RowsDeleted to determine how many rows were deleted during the last query operation. If RowsDeleted is -1, the query has not deleted any rows.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.17 RowsFetched Property

Used to get the number of the currently fetched rows.

Class

[TCustomIBCDataset](#)

Syntax

```
property RowsFetched: integer;
```

Remarks

Use the RowsFetched property to get the number of the currently fetched rows.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.18 RowsInserted Property

Used to indicate the number of rows that were inserted during the last query operation.

Class

[TCustomIBCDataset](#)

Syntax

```
property RowsInserted: integer;
```

Remarks

Check RowsInserted to determine how many rows were inserted during the last query operation. If RowsInserted is -1, the query has not inserted any rows.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.19 RowsUpdated Property

Used to indicate the number of rows that were updated during the last query operation.

Class

[TCustomIBCDataset](#)

Syntax

```
property RowsUpdated: integer;
```

Remarks

Check RowsUpdated to determine how many rows were updated during the last query operation. If RowsUpdated is -1, the query has not updated any rows.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.20 SmartFetch Property

The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.

Class

[TCustomIBCDataset](#)

Syntax

```
property SmartFetch: TSmartFetchOptions;
```

See Also

- [TSmartFetchOptions](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.21 SQLType Property

Used to get the typecode of the SQL statement being processed by the InterBase database server.

Class

[TCustomIBCDataset](#)

Syntax

```
property SQLType: integer;
```

Remarks

Read the SQLType property to get the typecode of the SQL statement being processed by

the InterBase database server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.22 Transaction Property

Used to determine the transaction under which the query of this dataset executes.

Class

[TCustomIBCDataset](#)

Syntax

```
property Transaction: TIBCTransaction stored IsTransactionStored;
```

Remarks

Use the Transaction property to determine the transaction under which the query of this dataset executes. You can separately set transaction for executing modifying queries with the [UpdateTransaction](#) property. By default the Transaction and the UpdateTransaction properties are the same.

See Also

- [UpdateTransaction](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.23 UpdateObject Property

Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.

Class

[TCustomIBCDataset](#)

Syntax

```
property UpdateObject: TIBCUpdateSQL;
```

Remarks

The UpdateObject property specifies an update object component which provides SQL statements that perform updates of the read-only datasets when cached updates are enabled.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.2.24 UpdateTransaction Property

Used to get or set the transaction for modifying a dataset.

Class

[TCustomIBCDataset](#)

Syntax

```
property UpdateTransaction: TIBCTransaction;
```

Remarks

Use the UpdateTransaction property to set or get the transaction for modifying a dataset. By default UpdateTransaction is the same as [Transaction](#).

See Also

- [Transaction](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.3 Methods

Methods of the **TCustomIBCDataset** class.

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

Public

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

AddWhere (inherited from TCustomDADataset)	Adds condition to the WHERE clause of SELECT statement in the SQL property.
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
BreakExec (inherited from TCustomDADataset)	Breaks execution of the SQL statement on the server.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
CreateBlobStream (inherited from TCustomDADataset)	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
CreateProcCall	Assigns PL/SQL block that calls stored procedure to the SQL property.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
DeleteWhere (inherited from TCustomDADataset)	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
Execute (inherited from TCustomDADataset)	Overloaded. Executes a SQL statement on the server.

Executing (inherited from TCustomDADataset)	Indicates whether SQL statement is still being executed.
Fetched (inherited from TCustomDADataset)	Used to find out whether TCustomDADataset has fetched all rows.
Fetching (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is still fetching rows.
FetchingAll (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is fetching all rows to the end.
FindKey (inherited from TCustomDADataset)	Searches for a record which contains specified field values.
FindMacro (inherited from TCustomDADataset)	Finds a macro with the specified name.
FindNearest (inherited from TCustomDADataset)	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.
FindParam	Determines if a parameter with the specified name exists in a dataset.
GetArray	Retrieves a TIBCArry object for a field when only its name is known.
GetBlob	Retrieves a TIBCBlob object for a field when only its name is known.
GetDataType (inherited from TCustomDADataset)	Returns internal field types defined in the MemData and accompanying modules.
GetFieldObject (inherited from TCustomDADataset)	Returns a multireference shared object from field.
GetFieldPrecision (inherited from TCustomDADataset)	Retrieves the precision of a number field.
GetFieldScale (inherited from TCustomDADataset)	Retrieves the scale of a number field.
GetKeyFieldNames (inherited from TCustomDADataset)	Provides a list of available key field names.

GetOrderBy (inherited from TCustomDADataset)	Retrieves an ORDER BY clause from a SQL statement.
GotoCurrent (inherited from TCustomDADataset)	Sets the current record in this dataset similar to the current record in another dataset.
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.
Lock (inherited from TCustomDADataset)	Locks the current record.
MacroByName (inherited from TCustomDADataset)	Finds a macro with the specified name.
ParamByName	Called to set or use parameter information for a specific parameter based on its name.
Prepare (inherited from TCustomDADataset)	Allocates, opens, and parses cursor for a query.
RefreshRecord (inherited from TCustomDADataset)	Actualizes field values for the current record.
RestoreSQL (inherited from TCustomDADataset)	Restores the SQL property modified by AddWhere and SetOrderBy.
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.
SaveSQL (inherited from TCustomDADataset)	Saves the SQL property value to BaseSQL.
SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

SetOrderBy (inherited from TCustomDADataset)	Builds an ORDER BY clause of a SELECT statement.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
SQLSaved (inherited from TCustomDADataset)	Determines if the SQL property value was saved to the BaseSQL property.
UnLock (inherited from TCustomDADataset)	Releases a record lock.
UnPrepare (inherited from TMemDataSet)	Frees the resources allocated for a previously prepared query on the server and client sides.
UpdateResult (inherited from TMemDataSet)	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
UpdateStatus (inherited from TMemDataSet)	Indicates the current update status for the dataset when cached updates are enabled.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.3.1 CreateProcCall Method

Assigns PL/SQL block that calls stored procedure to the SQL property.

Class

[TCustomIBCDataset](#)

Syntax

```
procedure CreateProcCall(const Name: string);
```

Parameters

Name

Holds the name of the stored procedure

Remarks

Call the CreateProcCall method to assign PL/SQL block that calls stored procedure specified by Name to the SQL property. Retrieves the information about parameters of the procedure from InterBase. After calling CreateProcCall you can execute stored procedure by Execute method.

See Also

- [TCustomDADataset.Execute](#)
- [TCustomDACConnection.ExecProc](#)
- [TIBCStoredProc](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.3.2 FindParam Method

Determines if a parameter with the specified name exists in a dataset.

Class

[TCustomIBCDataset](#)

Syntax

```
function FindParam(const value: string): TIBCParam;
```

Parameters

Value

Holds the name of the param for which to search.

Return Value

The TIBCPParam object for the specified Name, if a param with a matching name was found. Nil otherwise.

Remarks

Call the FindParam method to determine if a parameter with the specified name exists in a dataset. Name is the name of the param for which to search. If FindParam finds a param with a matching name, it returns the TIBCPParam object for the specified Name. Otherwise it returns nil.

See Also

- [TCustomDADDataSet.Params](#)
- [ParamByName](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.3.3 GetArray Method

Retrieves a TIBCArray object for a field when only its name is known.

Class

[TCustomIBCDataset](#)

Syntax

```
function GetArray(FieldDesc: TFieldDesc): TIBCArray;  
overload; function GetArray(const FieldName: string): TIBCArray;  
overload;
```

Parameters

FieldName

Holds the field name of an existing field.

Return Value

The TIBCArray object, if a field with a matching name was found.

Remarks

Call the `GetArray` method to retrieve a `TIBCArray` object for a field when only its name is known. `FieldName` is the name of an existing field. The field should have the `ftIBCArray` type.

See Also

- [TIBCArray](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.3.4 GetBlob Method

Retrieves a `TIBCBlob` object for a field when only its name is known.

Class

[TCustomIBCDataSet](#)

Syntax

```
function GetBlob(const FieldName: string): TIBCBlob;
```

Parameters

FieldName

Holds the field name of an existing field.

Return Value

The `TIBCBlob` object, if a field with a matching name was found.

Remarks

Call the `GetBlob` method to retrieve a `TIBCBlob` object for a field when only its name is known. `FieldName` is the name of an existing field. The field should have the `ftIBCBlob` type.

See Also

- [TIBCBlob](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.1.3.5 ParamByName Method

Called to set or use parameter information for a specific parameter based on its name.

Class

[TCustomIBCDataset](#)

Syntax

```
function ParamByName(const Value: string): TIBCPParam;
```

Parameters

Value

Holds the name of the parameter for which to retrieve information.

Return Value

A object, if there is a parameter with a matching name.

Remarks

Call the ParamByName method to set or use parameter information for a specific parameter based on its name. Name is the name of the parameter for which to retrieve information. ParamByName is used to set a parameter's value at runtime and returns a [TIBCPParam](#) object.

See Also

- [TIBCPParam](#)
- [TCustomDADataset.Params](#)
- [TCustomDADataset.FindParam](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.2 TCustomIBCQuery Class

A base class for defining functionality for descendant classes which access database using SQL statements.

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

Unit

[IBC](#)

Syntax

```
TCustomIBCQuery = class(TCustomIBCDataSet);
```

Remarks

TCustomIBCQuery is a base class that defines functionality for descendant classes which access database using SQL statements. Applications never use the TCustomIBCQuery objects directly. Instead they use descendants of TCustomIBCQuery, such as TIBCQuery, TIBCStoredProc and TIBCTable.

TCustomIBCQuery implements functionality to update database tables using DML SQL statements. Put SQL statements into SQLInsert, SQLDelete, SQLUpdate properties. There is no restriction on their syntax, so any SQL statements are allowed. Usually you need to use INSERT, DELETE and UPDATE statements but also you can use stored procedures in more diverse cases.

SQLInsert, SQLDelete, SQLUpdate, SQLLock, SQLRefresh properties support automatic binding of parameters which have names identical to fields captions. To retrieve the value of a field as it was before operation use the field name with 'OLD_'. This is especially useful when doing field comparisons in the WHERE clause of the statement. Use [TCustomDADDataSet.BeforeUpdateExecute](#) event to assign value to additional parameters and [TCustomDADDataSet.AfterUpdateExecute](#) event for reading them.

TCustomIBCQuery is read-only when none of SQLInsert, SQLDelete, SQLUpdate properties are defined.

Inheritance Hierarchy

[TMemDataSet](#)

[TCustomDADDataSet](#)

[TCustomIBCDataSet](#)

TCustomIBCQuery

See Also

- [TCustomIBCDataSet](#)
- [TIBCQuery](#)

- [TIBCStoredProc](#)
- [TIBCTable](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.2.1 Members

[TCustomIBCQuery](#) class overview.

Properties

Name	Description
AutoCommit (inherited from TCustomIBCDataSet)	Used to automatically commit each update, insert or delete statement by database server.
BaseSQL (inherited from TCustomDADataset)	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions (inherited from TCustomDADataset)	Used to add WHERE conditions to a query
Connection (inherited from TCustomIBCDataSet)	Used to specify the connection in which the dataset will be executed.
Cursor (inherited from TCustomIBCDataSet)	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
DataTypeMap (inherited from TCustomDADataset)	Used to set data type mapping rules
Debug (inherited from TCustomDADataset)	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields (inherited from TCustomDADataset)	Used to specify the fields that correspond to the

	foreign key fields from MasterFields when building master/detail relationship.
Disconnected (inherited from TCustomDADataset)	Used to keep dataset opened after connection is closed.
DMLRefresh (inherited from TCustomIBCDataset)	Used to refresh record by the RETURNING clause when insert is performed.
Encryption (inherited from TCustomIBCDataset)	Used to specify encryption options in a dataset.
ExplainPlan (inherited from TCustomIBCDataset)	Used to obtain the query execution plan for Table, Query, and SQL components.
FetchAll (inherited from TCustomIBCDataset)	Used to retrieve all records in a dataset.
FetchRows (inherited from TCustomDADataset)	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL (inherited from TCustomDADataset)	Used to change the WHERE clause of SELECT statement and reopen a query.
FinalSQL (inherited from TCustomDADataset)	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
GeneratorMode (inherited from TCustomIBCDataset)	Used to specify which method is used internally to generate a sequenced field.
GeneratorStep (inherited from TCustomIBCDataset)	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
Handle (inherited from TCustomIBCDataset)	Used to specify the handle for the SQL statement of TCustomIBCDataset.
IndexFieldNames (inherited from TMemDataset)	Used to get or set the list of fields on which the recordset is sorted.

IsQuery (inherited from TCustomIBCDataSet)	Used to check if the SQL statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
KeyFields (inherited from TCustomDADataset)	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
KeyGenerator (inherited from TCustomIBCDataSet)	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
LockMode (inherited from TCustomIBCDataSet)	Used to indicate when to perform a locking of editing record.
MacroCount (inherited from TCustomDADataset)	Used to get the number of macros associated with the Macros property.
Macros (inherited from TCustomDADataset)	Makes it possible to change SQL queries easily.
MasterFields (inherited from TCustomDADataset)	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)	Used to specify the data source component which binds current dataset to the master one.
Options (inherited from TCustomIBCDataSet)	Used to specify the behaviour of the

	TCustomIBCDataSet object.
ParamCheck (inherited from TCustomDADDataSet)	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
ParamCount (inherited from TCustomDADDataSet)	Used to indicate how many parameters are there in the Params property.
Params (inherited from TCustomDADDataSet)	Used to view and set parameter names, values, and data types dynamically.
Plan (inherited from TCustomIBCDataSet)	Used to get or set the PLAN clause of the SELECT statement.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
ReadOnly (inherited from TCustomDADDataSet)	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions (inherited from TCustomDADDataSet)	Used to indicate when the editing record is refreshed.
RowsAffected (inherited from TCustomDADDataSet)	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
RowsDeleted (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were deleted during the last query operation.
RowsFetched (inherited from TCustomIBCDataSet)	Used to get the number of the currently fetched rows.
RowsInserted (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were inserted during the last query operation.
RowsUpdated (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were updated during the last query operation.

SmartFetch (inherited from TCustomIBCDataSet)	The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.
SQL (inherited from TCustomDADataset)	Used to provide a SQL statement that a query component executes when its Open method is called.
SQLDelete (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying a deletion to a record.
SQLInsert (inherited from TCustomDADataset)	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
SQLLock (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to perform a record lock.
SQLRecCount (inherited from TCustomDADataset)	Used to specify the SQL statement that is used to get the record count when opening a dataset.
SQLRefresh (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.
SQLType (inherited from TCustomIBCDataSet)	Used to get the typecode of the SQL statement being processed by the InterBase database server.
SQLUpdate (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying an update to a dataset.
Transaction (inherited from TCustomIBCDataSet)	Used to determine the transaction under which the query of this dataset executes.
UniDirectional (inherited from TCustomDADataset)	Used if an application does not need bidirectional access to records in the

	result set.
UpdateObject (inherited from TCustomIBCDataset)	Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.
UpdateRecordTypes (inherited from TMemDataSet)	Used to indicate the update status for the current record when cached updates are enabled.
UpdatesPending (inherited from TMemDataSet)	Used to check the status of the cached updates buffer.
UpdateTransaction (inherited from TCustomIBCDataset)	Used to get or set the transaction for modifying a dataset.

Methods

Name	Description
AddWhere (inherited from TCustomDADataset)	Adds condition to the WHERE clause of SELECT statement in the SQL property.
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
BreakExec (inherited from TCustomDADataset)	Breaks execution of the SQL statement on the server.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
CreateBlobStream (inherited from TCustomDADataset)	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.

CreateProcCall (inherited from TCustomIBCDataSet)	Assigns PL/SQL block that calls stored procedure to the SQL property.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
DeleteWhere (inherited from TCustomDADDataSet)	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
Execute (inherited from TCustomDADDataSet)	Overloaded. Executes a SQL statement on the server.
Executing (inherited from TCustomDADDataSet)	Indicates whether SQL statement is still being executed.
Fetched (inherited from TCustomDADDataSet)	Used to find out whether TCustomDADDataSet has fetched all rows.
Fetching (inherited from TCustomDADDataSet)	Used to learn whether TCustomDADDataSet is still fetching rows.
FetchingAll (inherited from TCustomDADDataSet)	Used to learn whether TCustomDADDataSet is fetching all rows to the end.
FindKey (inherited from TCustomDADDataSet)	Searches for a record which contains specified field values.
FindMacro (inherited from TCustomDADDataSet)	Finds a macro with the specified name.
FindNearest (inherited from TCustomDADDataSet)	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.
FindParam (inherited from TCustomIBCDataSet)	Determines if a parameter with the specified name exists in a dataset.

GetArray (inherited from TCustomIBCDataSet)	Retrieves a TIBCArry object for a field when only its name is known.
GetBlob (inherited from TCustomIBCDataSet)	Retrieves a TIBCBlob object for a field when only its name is known.
GetDataType (inherited from TCustomDADataset)	Returns internal field types defined in the MemData and accompanying modules.
GetFieldObject (inherited from TCustomDADataset)	Returns a multireference shared object from field.
GetFieldPrecision (inherited from TCustomDADataset)	Retrieves the precision of a number field.
GetFieldScale (inherited from TCustomDADataset)	Retrieves the scale of a number field.
GetKeyFieldNames (inherited from TCustomDADataset)	Provides a list of available key field names.
GetOrderBy (inherited from TCustomDADataset)	Retrieves an ORDER BY clause from a SQL statement.
GotoCurrent (inherited from TCustomDADataset)	Sets the current record in this dataset similar to the current record in another dataset.
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.
Lock (inherited from TCustomDADataset)	Locks the current record.
MacroByName (inherited from TCustomDADataset)	Finds a macro with the specified name.
ParamByName (inherited from TCustomIBCDataSet)	Called to set or use parameter information for a specific parameter based on its name.
Prepare (inherited from TCustomDADataset)	Allocates, opens, and parses cursor for a query.

RefreshRecord (inherited from TCustomDADataset)	Actualizes field values for the current record.
RestoreSQL (inherited from TCustomDADataset)	Restores the SQL property modified by AddWhere and SetOrderBy.
RestoreUpdates (inherited from TMemDataSet)	Marks all records in the cache of updates as unapplied.
RevertRecord (inherited from TMemDataSet)	Cancel changes made to the current record when cached updates are enabled.
SaveSQL (inherited from TCustomDADataset)	Saves the SQL property value to BaseSQL.
SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
SetOrderBy (inherited from TCustomDADataset)	Builds an ORDER BY clause of a SELECT statement.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
SQLSaved (inherited from TCustomDADataset)	Determines if the SQL property value was saved to the BaseSQL property.
UnLock (inherited from TCustomDADataset)	Releases a record lock.
UnPrepare (inherited from TMemDataSet)	Frees the resources allocated for a previously prepared query on the server and client sides.
UpdateResult (inherited from TMemDataSet)	Reads the status of the latest call to the

	ApplyUpdates method while cached updates are enabled.
UpdateStatus (inherited from TMemDataSet)	Indicates the current update status for the dataset when cached updates are enabled.

Events

Name	Description
AfterExecute (inherited from TCustomDADataset)	Occurs after a component has executed a query to database.
AfterFetch (inherited from TCustomDADataset)	Occurs after dataset finishes fetching data from server.
AfterUpdateExecute (inherited from TCustomDADataset)	Occurs after executing insert, delete, update, lock and refresh operations.
BeforeFetch (inherited from TCustomDADataset)	Occurs before dataset is going to fetch block of records from the server.
BeforeUpdateExecute (inherited from TCustomDADataset)	Occurs before executing insert, delete, update, lock, and refresh operations.
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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.3 TCustomIBCTable Class

A base class that defines functionality for descendant classes which access data in a single table without writing SQL statements.

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

Unit

[IBC](#)

Syntax

```
TCustomIBCTable = class(TCustomIBCQuery);
```

Remarks

TCustomIBCTable component is inherited from TCustomIBCQuery class. Applications never use TCustomIBCTable objects directly. Instead they use descendants of TCustomIBCTable such as TIBCTable.

It allows to retrieve and update data in single table without writing SQL statements. Use TableName to specify the name of a table. TIBCTable uses KeyFields property to build SQL statements for updating table data. KeyFields is a string containing a semicolon-delimited list of field names. If KeyFields is not defined before opening, TIBCTable uses primary or unique key.

Inheritance Hierarchy

[TMemDataSet](#)

[TCustomDADataset](#)

[TCustomIBCDataSet](#)

[TCustomIBCQuery](#)

TCustomIBCTable

See Also

- [TIBCTable](#)
- [TCustomIBCDataSet](#)
- [TIBCQuery](#)
- [Master/Detail Relationships](#)
- [Updating Data with IBDAC Dataset Components](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.3.1 Members

[TCustomBCTable](#) class overview.

Properties

Name	Description
AutoCommit (inherited from TCustomIBCDataSet)	Used to automatically commit each update, insert or delete statement by database server.
BaseSQL (inherited from TCustomDADataset)	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions (inherited from TCustomDADataset)	Used to add WHERE conditions to a query
Connection (inherited from TCustomIBCDataSet)	Used to specify the connection in which the dataset will be executed.
Cursor (inherited from TCustomIBCDataSet)	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
DataTypeMap (inherited from TCustomDADataset)	Used to set data type mapping rules
Debug (inherited from TCustomDADataset)	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields (inherited from TCustomDADataset)	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
Disconnected (inherited from TCustomDADataset)	Used to keep dataset opened after connection is closed.
DMLRefresh (inherited from TCustomIBCDataSet)	Used to refresh record by the RETURNING clause

	when insert is performed.
Encryption (inherited from TCustomIBCDataSet)	Used to specify encryption options in a dataset.
Exists	Indicates whether a table with the name passed in TableName exists in the database.
ExplainPlan (inherited from TCustomIBCDataSet)	Used to obtain the query execution plan for Table, Query, and SQL components.
FetchAll (inherited from TCustomIBCDataSet)	Used to retrieve all records in a dataset.
FetchRows (inherited from TCustomDADataset)	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL (inherited from TCustomDADataset)	Used to change the WHERE clause of SELECT statement and reopen a query.
FinalSQL (inherited from TCustomDADataset)	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
GeneratorMode (inherited from TCustomIBCDataSet)	Used to specify which method is used internally to generate a sequenced field.
GeneratorStep (inherited from TCustomIBCDataSet)	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
Handle (inherited from TCustomIBCDataSet)	Used to specify the handle for the SQL statement of TCustomIBCDataSet.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
IsQuery (inherited from TCustomIBCDataSet)	Used to check if the SQL statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a

	range.
KeyFields (inherited from TCustomDADataset)	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
KeyGenerator (inherited from TCustomIBCDataSet)	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
LockMode (inherited from TCustomIBCDataSet)	Used to indicate when to perform a locking of editing record.
MacroCount (inherited from TCustomDADataset)	Used to get the number of macros associated with the Macros property.
Macros (inherited from TCustomDADataset)	Makes it possible to change SQL queries easily.
MasterFields (inherited from TCustomDADataset)	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)	Used to specify the data source component which binds current dataset to the master one.
Options (inherited from TCustomIBCDataSet)	Used to specify the behaviour of the TCustomIBCDataSet object.
ParamCheck (inherited from TCustomDADataset)	Used to specify whether parameters for the Params property are generated

	automatically after the SQL property was changed.
ParamCount (inherited from TCustomDADataset)	Used to indicate how many parameters are there in the Params property.
Params (inherited from TCustomDADataset)	Used to view and set parameter names, values, and data types dynamically.
Plan (inherited from TCustomIBCDataset)	Used to get or set the PLAN clause of the SELECT statement.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
ReadOnly (inherited from TCustomDADataset)	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions (inherited from TCustomDADataset)	Used to indicate when the editing record is refreshed.
RowsAffected (inherited from TCustomDADataset)	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
RowsDeleted (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were deleted during the last query operation.
RowsFetched (inherited from TCustomIBCDataset)	Used to get the number of the currently fetched rows.
RowsInserted (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were inserted during the last query operation.
RowsUpdated (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were updated during the last query operation.
SmartFetch (inherited from TCustomIBCDataset)	The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.

SQL (inherited from TCustomDADataset)	Used to provide a SQL statement that a query component executes when its Open method is called.
SQLDelete (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying a deletion to a record.
SQLInsert (inherited from TCustomDADataset)	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
SQLLock (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to perform a record lock.
SQLRecCount (inherited from TCustomDADataset)	Used to specify the SQL statement that is used to get the record count when opening a dataset.
SQLRefresh (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.
SQLType (inherited from TCustomIBCDataSet)	Used to get the typecode of the SQL statement being processed by the InterBase database server.
SQLUpdate (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying an update to a dataset.
Transaction (inherited from TCustomIBCDataSet)	Used to determine the transaction under which the query of this dataset executes.
UniDirectional (inherited from TCustomDADataset)	Used if an application does not need bidirectional access to records in the result set.
UpdateObject (inherited from TCustomIBCDataSet)	Used to specify an update object component which provides SQL statements that perform updates of the

	read-only datasets.
UpdateRecordTypes (inherited from TMemDataSet)	Used to indicate the update status for the current record when cached updates are enabled.
UpdatesPending (inherited from TMemDataSet)	Used to check the status of the cached updates buffer.
UpdateTransaction (inherited from TCustomIBCDataset)	Used to get or set the transaction for modifying a dataset.

Methods

Name	Description
AddWhere (inherited from TCustomDADataset)	Adds condition to the WHERE clause of SELECT statement in the SQL property.
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
BreakExec (inherited from TCustomDADataset)	Breaks execution of the SQL statement on the server.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
CreateBlobStream (inherited from TCustomDADataset)	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
CreateProcCall (inherited from TCustomIBCDataset)	Assigns PL/SQL block that calls stored procedure to the SQL property.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes

	to the database server.
DeleteTable	Deletes a table from a database.
DeleteWhere (inherited from TCustomDADataset)	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
EmptyTable	Truncates the current table content on the server.
Execute (inherited from TCustomDADataset)	Overloaded. Executes a SQL statement on the server.
Executing (inherited from TCustomDADataset)	Indicates whether SQL statement is still being executed.
Fetched (inherited from TCustomDADataset)	Used to find out whether TCustomDADataset has fetched all rows.
Fetching (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is still fetching rows.
FetchingAll (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is fetching all rows to the end.
FindKey (inherited from TCustomDADataset)	Searches for a record which contains specified field values.
FindMacro (inherited from TCustomDADataset)	Finds a macro with the specified name.
FindNearest (inherited from TCustomDADataset)	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.
FindParam (inherited from TCustomIBCDataSet)	Determines if a parameter with the specified name exists in a dataset.

GetArray (inherited from TCustomIBCDataSet)	Retrieves a TIBCArray object for a field when only its name is known.
GetBlob (inherited from TCustomIBCDataSet)	Retrieves a TIBCBlob object for a field when only its name is known.
GetDataType (inherited from TCustomDADataset)	Returns internal field types defined in the MemData and accompanying modules.
GetFieldObject (inherited from TCustomDADataset)	Returns a multireference shared object from field.
GetFieldPrecision (inherited from TCustomDADataset)	Retrieves the precision of a number field.
GetFieldScale (inherited from TCustomDADataset)	Retrieves the scale of a number field.
GetKeyFieldNames (inherited from TCustomDADataset)	Provides a list of available key field names.
GetOrderBy (inherited from TCustomDADataset)	Retrieves an ORDER BY clause from a SQL statement.
GotoCurrent (inherited from TCustomDADataset)	Sets the current record in this dataset similar to the current record in another dataset.
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.
Lock (inherited from TCustomDADataset)	Locks the current record.
MacroByName (inherited from TCustomDADataset)	Finds a macro with the specified name.
ParamByName (inherited from TCustomIBCDataSet)	Called to set or use parameter information for a specific parameter based on its name.
Prepare (inherited from TCustomDADataset)	Allocates, opens, and parses cursor for a query.

RefreshRecord (inherited from TCustomDADataset)	Actualizes field values for the current record.
RestoreSQL (inherited from TCustomDADataset)	Restores the SQL property modified by AddWhere and SetOrderBy.
RestoreUpdates (inherited from TMemDataSet)	Marks all records in the cache of updates as unapplied.
RevertRecord (inherited from TMemDataSet)	Cancel changes made to the current record when cached updates are enabled.
SaveSQL (inherited from TCustomDADataset)	Saves the SQL property value to BaseSQL.
SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
SetOrderBy (inherited from TCustomDADataset)	Builds an ORDER BY clause of a SELECT statement.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
SQLSaved (inherited from TCustomDADataset)	Determines if the SQL property value was saved to the BaseSQL property.
UnLock (inherited from TCustomDADataset)	Releases a record lock.
UnPrepare (inherited from TMemDataSet)	Frees the resources allocated for a previously prepared query on the server and client sides.
UpdateResult (inherited from TMemDataSet)	Reads the status of the latest call to the

	ApplyUpdates method while cached updates are enabled.
UpdateStatus (inherited from TMemDataSet)	Indicates the current update status for the dataset when cached updates are enabled.

Events

Name	Description
AfterExecute (inherited from TCustomDADataset)	Occurs after a component has executed a query to database.
AfterFetch (inherited from TCustomDADataset)	Occurs after dataset finishes fetching data from server.
AfterUpdateExecute (inherited from TCustomDADataset)	Occurs after executing insert, delete, update, lock and refresh operations.
BeforeFetch (inherited from TCustomDADataset)	Occurs before dataset is going to fetch block of records from the server.
BeforeUpdateExecute (inherited from TCustomDADataset)	Occurs before executing insert, delete, update, lock, and refresh operations.
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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.3.2 Properties

Properties of the **TCustomIBCTable** class.

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

Public

Name	Description
AutoCommit (inherited from TCustomIBCDataSet)	Used to automatically commit each update, insert or delete statement by database server.
BaseSQL (inherited from TCustomDADataset)	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions (inherited from TCustomDADataset)	Used to add WHERE conditions to a query
Connection (inherited from TCustomIBCDataSet)	Used to specify the connection in which the dataset will be executed.
Cursor (inherited from TCustomIBCDataSet)	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
DataTypeMap (inherited from TCustomDADataset)	Used to set data type mapping rules
Debug (inherited from TCustomDADataset)	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields (inherited from TCustomDADataset)	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
Disconnected (inherited from TCustomDADataset)	Used to keep dataset opened after connection is closed.
DMLRefresh (inherited from TCustomIBCDataSet)	Used to refresh record by the RETURNING clause when insert is performed.
Encryption (inherited from TCustomIBCDataSet)	Used to specify encryption options in a dataset.

Exists	Indicates whether a table with the name passed in TableName exists in the database.
ExplainPlan (inherited from TCustomIBCDataSet)	Used to obtain the query execution plan for Table, Query, and SQL components.
FetchAll (inherited from TCustomIBCDataSet)	Used to retrieve all records in a dataset.
FetchRows (inherited from TCustomDADataset)	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL (inherited from TCustomDADataset)	Used to change the WHERE clause of SELECT statement and reopen a query.
FinalSQL (inherited from TCustomDADataset)	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
GeneratorMode (inherited from TCustomIBCDataSet)	Used to specify which method is used internally to generate a sequenced field.
GeneratorStep (inherited from TCustomIBCDataSet)	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
Handle (inherited from TCustomIBCDataSet)	Used to specify the handle for the SQL statement of TCustomIBCDataSet.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
IsQuery (inherited from TCustomIBCDataSet)	Used to check if the SQL statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
KeyFields (inherited from TCustomDADataset)	Used to build SQL statements for the

	SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
KeyGenerator (inherited from TCustomIBCDataSet)	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
LockMode (inherited from TCustomIBCDataSet)	Used to indicate when to perform a locking of editing record.
MacroCount (inherited from TCustomDADataset)	Used to get the number of macros associated with the Macros property.
Macros (inherited from TCustomDADataset)	Makes it possible to change SQL queries easily.
MasterFields (inherited from TCustomDADataset)	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)	Used to specify the data source component which binds current dataset to the master one.
Options (inherited from TCustomIBCDataSet)	Used to specify the behaviour of the TCustomIBCDataSet object.
ParamCheck (inherited from TCustomDADataset)	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.

ParamCount (inherited from TCustomDADataset)	Used to indicate how many parameters are there in the Params property.
Params (inherited from TCustomDADataset)	Used to view and set parameter names, values, and data types dynamically.
Plan (inherited from TCustomIBCDataset)	Used to get or set the PLAN clause of the SELECT statement.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
ReadOnly (inherited from TCustomDADataset)	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions (inherited from TCustomDADataset)	Used to indicate when the editing record is refreshed.
RowsAffected (inherited from TCustomDADataset)	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
RowsDeleted (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were deleted during the last query operation.
RowsFetched (inherited from TCustomIBCDataset)	Used to get the number of the currently fetched rows.
RowsInserted (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were inserted during the last query operation.
RowsUpdated (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were updated during the last query operation.
SmartFetch (inherited from TCustomIBCDataset)	The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.
SQL (inherited from TCustomDADataset)	Used to provide a SQL statement that a query

	component executes when its Open method is called.
SQLDelete (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying a deletion to a record.
SQLInsert (inherited from TCustomDADataset)	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
SQLLock (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to perform a record lock.
SQLRecCount (inherited from TCustomDADataset)	Used to specify the SQL statement that is used to get the record count when opening a dataset.
SQLRefresh (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.
SQLType (inherited from TCustomIBCDataSet)	Used to get the typecode of the SQL statement being processed by the InterBase database server.
SQLUpdate (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying an update to a dataset.
Transaction (inherited from TCustomIBCDataSet)	Used to determine the transaction under which the query of this dataset executes.
UniDirectional (inherited from TCustomDADataset)	Used if an application does not need bidirectional access to records in the result set.
UpdateObject (inherited from TCustomIBCDataSet)	Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.

UpdateRecordTypes (inherited from TMemDataSet)	Used to indicate the update status for the current record when cached updates are enabled.
UpdatesPending (inherited from TMemDataSet)	Used to check the status of the cached updates buffer.
UpdateTransaction (inherited from TCustomIBDataSet)	Used to get or set the transaction for modifying a dataset.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.3.2.1 Exists Property

Indicates whether a table with the name passed in TableName exists in the database.

Class

[TCustomIBCTable](#)

Syntax

```
property Exists: Boolean;
```

Remarks

Use the Exists property to determine whether a table with the name passed in TableName exists in the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.3.3 Methods

Methods of the **TCustomIBCTable** class.

For a complete list of the **TCustomIBCTable** class members, see the [TCustomIBCTable](#)

[Members](#) topic.

Public

Name	Description
AddWhere (inherited from TCustomDADataset)	Adds condition to the WHERE clause of SELECT statement in the SQL property.
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
BreakExec (inherited from TCustomDADataset)	Breaks execution of the SQL statement on the server.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
CreateBlobStream (inherited from TCustomDADataset)	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
CreateProcCall (inherited from TCustomIBCDataset)	Assigns PL/SQL block that calls stored procedure to the SQL property.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
DeleteTable	Deletes a table from a database.
DeleteWhere (inherited from TCustomDADataset)	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.

EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
EmptyTable	Truncates the current table content on the server.
Execute (inherited from TCustomDADataset)	Overloaded. Executes a SQL statement on the server.
Executing (inherited from TCustomDADataset)	Indicates whether SQL statement is still being executed.
Fetched (inherited from TCustomDADataset)	Used to find out whether TCustomDADataset has fetched all rows.
Fetching (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is still fetching rows.
FetchingAll (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is fetching all rows to the end.
FindKey (inherited from TCustomDADataset)	Searches for a record which contains specified field values.
FindMacro (inherited from TCustomDADataset)	Finds a macro with the specified name.
FindNearest (inherited from TCustomDADataset)	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.
FindParam (inherited from TCustomIBCDataSet)	Determines if a parameter with the specified name exists in a dataset.
GetArray (inherited from TCustomIBCDataSet)	Retrieves a TIBCArray object for a field when only its name is known.
GetBlob (inherited from TCustomIBCDataSet)	Retrieves a TIBCBlob object for a field when only its name is known.
GetDataType (inherited from TCustomDADataset)	Returns internal field types defined in the MemData and accompanying modules.

GetFieldObject (inherited from TCustomDADataset)	Returns a multireference shared object from field.
GetFieldPrecision (inherited from TCustomDADataset)	Retrieves the precision of a number field.
GetFieldScale (inherited from TCustomDADataset)	Retrieves the scale of a number field.
GetKeyFieldNames (inherited from TCustomDADataset)	Provides a list of available key field names.
GetOrderBy (inherited from TCustomDADataset)	Retrieves an ORDER BY clause from a SQL statement.
GotoCurrent (inherited from TCustomDADataset)	Sets the current record in this dataset similar to the current record in another dataset.
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.
Lock (inherited from TCustomDADataset)	Locks the current record.
MacroByName (inherited from TCustomDADataset)	Finds a macro with the specified name.
ParamByName (inherited from TCustomIBCDataSet)	Called to set or use parameter information for a specific parameter based on its name.
Prepare (inherited from TCustomDADataset)	Allocates, opens, and parses cursor for a query.
RefreshRecord (inherited from TCustomDADataset)	Actualizes field values for the current record.
RestoreSQL (inherited from TCustomDADataset)	Restores the SQL property modified by AddWhere and SetOrderBy.
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.
SaveSQL (inherited from TCustomDADataset)	Saves the SQL property value to BaseSQL.
SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
SetOrderBy (inherited from TCustomDADataset)	Builds an ORDER BY clause of a SELECT statement.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
SQLSaved (inherited from TCustomDADataset)	Determines if the SQL property value was saved to the BaseSQL property.
UnLock (inherited from TCustomDADataset)	Releases a record lock.
UnPrepare (inherited from TMemDataSet)	Frees the resources allocated for a previously prepared query on the server and client sides.
UpdateResult (inherited from TMemDataSet)	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
UpdateStatus (inherited from TMemDataSet)	Indicates the current update status for the dataset when cached updates are enabled.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.3.3.1 DeleteTable Method

Deletes a table from a database.

Class

[TCustomIBCTable](#)

Syntax

```
procedure DeleteTable;
```

Remarks

Call the DeleteTable method to delete a table from database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.3.3.2 EmptyTable Method

Truncates the current table content on the server.

Class

[TCustomIBCTable](#)

Syntax

```
procedure EmptyTable;
```

Remarks

Call the EmptyTable method to truncate the current table content on the server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.11.1.4 TIBCArray Class

A class representing the value of the InterBase array data type.

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

Unit

[IBC](#)

Syntax

```
TIBCArray = class (TCustomIBCArray);
```

Remarks

TIBCArray represents the value of the InterBase array data type. You can get a TIBCArray object by the TCustomIBCDataset.GetArray method after fetching rows contained in an array field. You should explicitly add the T:Devart.IbDac.Units.IBCArray unit to 'uses' list to use the TIBCArray class.

Inheritance Hierarchy

[TSharedObject](#)

[TDBObject](#)

[TCustomIBCArray](#)

TIBCArray

See Also

- [TCustomIBCDataset.GetArray](#)
- [TIBCPParam.AsArray](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.4.1 Members

[TIBCArray](#) class overview.

Properties

Name	Description
ArrayDimensions (inherited from TCustomIBCArray)	Contains the array dimensions count.
ArrayHighBound (inherited from TCustomIBCArray)	Used to get or set the upper boundary of the defined dimension subscript.
ArrayID (inherited from TCustomIBCArray)	Contains the array ID.
ArrayLowBound (inherited from TCustomIBCArray)	Used to get or set the lower boundary of the defined dimension subscript.
ArraySize (inherited from TCustomIBCArray)	Used to determine the size of the whole array data in bytes.
AsString (inherited from TCustomIBCArray)	Used to return array as string.
Cached (inherited from TCustomIBCArray)	Indicates whether to cache array data on the client side.
CachedDimensions (inherited from TCustomIBCArray)	Contains cached array dimensions count.
CachedHighBound (inherited from TCustomIBCArray)	Used to get the upper boundary of defined dimension subscript of cached array elements.
CachedLowBound (inherited from TCustomIBCArray)	Used to get the lower boundary of the defined dimension subscript of cached array elements.
CachedSize (inherited from TCustomIBCArray)	Used to get the cached array data size in bytes.
ColumnName (inherited from TCustomIBCArray)	Used to get or set the name of the table column that has array type.
DbHandle (inherited from TCustomIBCArray)	Contains the handle of a database where the array is stored.
IsNull (inherited from TCustomIBCArray)	Used to define whether the array field in the database is null.
Items (inherited from TCustomIBCArray)	Used to get or set array items.
ItemScale (inherited from TCustomIBCArray)	Used to get or set the scale for array items for the NUMERIC and DECIMAL

	datatypes.
ItemSize (inherited from TCustomIBArray)	Contains the size of an array item.
ItemType	Used to return the type of an array element.
Modified (inherited from TCustomIBArray)	Used to indicate if the modifications done in cache were saved to the database.
RefCount (inherited from TSharedObject)	Used to return the count of reference to a TSharedObject object.
TableName (inherited from TCustomIBArray)	Used to set the name of the table containing an array field.
TrHandle (inherited from TCustomIBArray)	Contains the handle of the transaction in which the array is read or written.

Methods

Name	Description
AddRef (inherited from TSharedObject)	Increments the reference count for the number of references dependent on the TSharedObject object.
Assign (inherited from TCustomIBArray)	Copies the Source object content to the current one.
ClearArray (inherited from TCustomIBArray)	Clears all array values on the server if Cached is set to False.
CreateTemporaryArray (inherited from TCustomIBArray)	Creates a temporary array on the InterBase server.
GetArrayInfo (inherited from TCustomIBArray)	Used to get array descriptor.
GetItemAsDateTime (inherited from TCustomIBArray)	Reads the value of an array item into an object or variable of the TDateTime type.
GetItemAsFloat (inherited from TCustomIBArray)	Reads the value of an array item into a floating-point number.
GetItemAsInteger (inherited from TCustomIBArray)	Reads the value of an array item into a integer.

GetItemAsSmallInt (inherited from TCustomIBCArray)	Reads the value of an array item into a short integer.
GetItemAsString (inherited from TCustomIBCArray)	Reads the value of an array item into a string.
GetItemAsWideString (inherited from TCustomIBCArray)	Reads the value of an array item into a WideString.
GetItemsSlice (inherited from TCustomIBCArray)	Returns the array slice items' values.
GetItemValue (inherited from TCustomIBCArray)	Returns the array item value.
ReadArray (inherited from TCustomIBCArray)	Reads an array from the database to memory.
ReadArrayItem (inherited from TCustomIBCArray)	Reads the array item specified by indices from the database to memory.
ReadArraySlice (inherited from TCustomIBCArray)	Reads array slice from the database to memory.
Release (inherited from TSharedObject)	Decrements the reference count.
SetItemAsDateTime (inherited from TCustomIBCArray)	Assigns the TDateTime value to the contents of an array item.
SetItemAsFloat (inherited from TCustomIBCArray)	Assigns floating-point value to the contents of an array item.
SetItemAsInteger (inherited from TCustomIBCArray)	Assigns integer value to the contents of an array item.
SetItemAsSmallInt (inherited from TCustomIBCArray)	Assigns short integer value to the contents of an array item.
SetItemAsString (inherited from TCustomIBCArray)	Assigns string value to the contents of an array item.
SetItemAsWideString (inherited from TCustomIBCArray)	Assigns WideString value to the contents of an array item.
SetItemsSlice (inherited from TCustomIBCArray)	Sets the array slice values.
SetItemValue (inherited from TCustomIBCArray)	Sets the array item value.
WriteArray (inherited from TCustomIBCArray)	Writes all cached array values to the database.
WriteArraySlice (inherited from TCustomIBCArray)	Writes cached array slice.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.11.1.4.2 Properties

Properties of the **TIBCArray** class.

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

Public

Name	Description
ArrayDimensions (inherited from TCustomIBCArray)	Contains the array dimensions count.
ArrayHighBound (inherited from TCustomIBCArray)	Used to get or set the upper boundary of the defined dimension subscript.
ArrayID (inherited from TCustomIBCArray)	Contains the array ID.
ArrayLowBound (inherited from TCustomIBCArray)	Used to get or set the lower boundary of the defined dimension subscript.
ArraySize (inherited from TCustomIBCArray)	Used to determine the size of the whole array data in bytes.
AsString (inherited from TCustomIBCArray)	Used to return array as string.
Cached (inherited from TCustomIBCArray)	Indicates whether to cache array data on the client side.
CachedDimensions (inherited from TCustomIBCArray)	Contains cached array dimensions count.
CachedHighBound (inherited from TCustomIBCArray)	Used to get the upper boundary of defined dimension subscript of cached array elements.
CachedLowBound (inherited from TCustomIBCArray)	Used to get the lower boundary of the defined dimension subscript of cached array elements.
CachedSize (inherited from TCustomIBCArray)	Used to get the cached array data size in bytes.
ColumnName (inherited from TCustomIBCArray)	Used to get or set the name of the table column that has array type.

DbHandle (inherited from TCustomIBCArray)	Contains the handle of a database where the array is stored.
IsNull (inherited from TCustomIBCArray)	Used to define whether the array field in the database is null.
Items (inherited from TCustomIBCArray)	Used to get or set array items.
ItemScale (inherited from TCustomIBCArray)	Used to get or set the scale for array items for the NUMERIC and DECIMAL datatypes.
ItemSize (inherited from TCustomIBCArray)	Contains the size of an array item.
ItemType	Used to return the type of an array element.
Modified (inherited from TCustomIBCArray)	Used to indicate if the modifications done in cache were saved to the database.
RefCount (inherited from TSharedObject)	Used to return the count of reference to a TSharedObject object.
TableName (inherited from TCustomIBCArray)	Used to set the name of the table containing an array field.
TrHandle (inherited from TCustomIBCArray)	Contains the handle of the transaction in which the array is read or written.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.4.2.1 ItemType Property

Used to return the type of an array element.

Class

[TIBCArray](#)

Syntax

```
property ItemType: TFieldType;
```

Remarks

Read the ItemType property to return the type of an array element.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.5 TIBCArrayField Class

A class encapsulating the fundamental behavior common to the InterBase array fields.

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

Unit

[IBC](#)

Syntax

```
TIBCArrayField = class(TFieldType);
```

Remarks

TIBCArrayField encapsulates the fundamental behavior common to the InterBase array fields.

The InterBase array fields can contain multiple data items of the same datatypes.

TIBCArrayField introduces new AsArray property for accessing array data.

See Also

- [TIBCArray](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.5.1 Members

[TIBCArrayField](#) class overview.

Properties

Name	Description
AsArray	Used to specify the value of the field when it represents the value of the InterBase array type.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.5.2 Properties

Properties of the **TIBCArrayField** class.

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

Public

Name	Description
AsArray	Used to specify the value of the field when it represents the value of the InterBase array type.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.5.2.1 AsArray Property

Used to specify the value of the field when it represents the value of the InterBase array type.

Class

[TIBCArrayField](#)

Syntax

```
property AsArray: TIBCArray;
```

Remarks

Use the AsArray property to specify the value of the field when it represents the value of the InterBase array type.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6 TIBConnection Class

A component for setting and controlling connections to an InterBase database.

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

Unit

[IBC](#)

Syntax

```
TIBConnection = class(TCustomDAConnection);
```

Remarks

The TIBConnection component is used to maintain connection to the InterBase database. After setting the Username, Password, and Database properties, you can establish a connection to the database by calling the Open method or setting the Connected property to True.

The TIBConnection component contains internal transaction component that is accessible through the [TIBConnection.DefaultTransaction](#) property. It is possible to create applications without manual adding TIBCTransaction components. But you can use external transaction components instead of internal transaction. To do it create a TIBCTransaction component and assign the TIBConnection.DefaultTransaction property to this transaction. If you need to restore internal transaction - just reset the TIBConnection.DefaultTransaction property to nil.

All components which are dedicated to perform data access, such as TIBCQuery, TIBCSQL, TIBCScript, must have their Connection property assigned with one of the TIBConnection

instances.

Inheritance Hierarchy

[TCustomDACConnection](#)

TIBCCConnection

See Also

- [TIBCCConnection.DefaultTransaction](#)
- [TCustomIBCDataSet.Connection](#)
- [TIBCSQL.Connection](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.1 Members

[TIBCCConnection](#) class overview.

Properties

Name	Description
AutoCommit	Used to permit or prevent permanent updates, insertions, and deletions of data associated with the current transaction against the database server.
ClientLibrary	Used to set or get the client library location.
ConnectDialog (inherited from TCustomDACConnection)	Allows to link a TCustomConnectDialog component.
Connected	Used to establish a database connection.
ConnectPrompt	Used to provide login support.
ConnectionString (inherited from TCustomDACConnection)	Used to specify the connection information, such as: UserName, Password, Server, etc.

ConvertEOL (inherited from TCustomDACConnection)	Allows customizing line breaks in string fields and parameters.
Database	Used to set the name of the database to associate with TIBCCConnection component.
DatabaseInfo	Used to get information about the connected database.
DBSQLDialect	Shows the database SQL dialect.
Debug	Used to display SQL statements being executed with their parameter values and data types.
DefaultTransaction	Used to access default database connection transaction.
Handle	Used to make calls directly to the InterBase API.
InTransaction (inherited from TCustomDACConnection)	Indicates whether the transaction is active.
LastError	Used to get the error code which resulted from the previous call to the InterBase server.
LoginPrompt (inherited from TCustomDACConnection)	Specifies whether a login dialog appears immediately before opening a new connection.
Options	Used to specify the behaviour of the TIBCCConnection object.
Params	Used to specify the connection parameters.
Password	Used to specify the password for a connection.
Pooling (inherited from TCustomDACConnection)	Enables or disables using connection pool.
PoolingOptions (inherited from TCustomDACConnection)	Specifies the behaviour of connection pool.

Port	Used to specify the port number for connections to the server.
Server	Used to supply the server name to handle server's request for a login.
SQL	Used to execute any SQL statement.
SQLDialect	Used to set or return SQL Dialect used by InterBase client.
SSLOptions	Used to set up the SSL options.
TransactionCount	Used to return the number of transactions currently associated with this TIBCConnection component.
Transactions	Used to specify a transaction for the given index.
Username	Used to provide a user name.

Methods

Name	Description
AddTransaction	Associates a TIBCTransaction component with the database component.
ApplyUpdates (inherited from TCustomDAConnection)	Overloaded. Applies changes in datasets.
AssignConnect	Shares database connection between the TIBCConnection components.
Commit (inherited from TCustomDAConnection)	Commits current transaction.
CommitRetaining	Stores to the database server all changes of data associated with the default database transaction permanently and then retains

	the transaction context.
Connect (inherited from TCustomDACConnection)	Establishes a connection to the server.
CreateDatabase	Creates an InterBase database.
CreateSQL (inherited from TCustomDACConnection)	Creates a component for queries execution.
Disconnect (inherited from TCustomDACConnection)	Performs disconnect.
DropDatabase	Used to drop database and remove database file from server.
ExecProc (inherited from TCustomDACConnection)	Allows to execute stored procedure or function providing its name and parameters.
ExecProcEx (inherited from TCustomDACConnection)	Allows to execute a stored procedure or function.
ExecSQL (inherited from TCustomDACConnection)	Executes a SQL statement with parameters.
ExecSQLEx (inherited from TCustomDACConnection)	Executes any SQL statement outside the TQuery or TSQL components.
FindDefaultTransaction	Returns the default transaction for the database.
GetDatabaseNames (inherited from TCustomDACConnection)	Returns a database list from the server.
GetKeyFieldNames (inherited from TCustomDACConnection)	Provides a list of available key field names.
GetStoredProcNames (inherited from TCustomDACConnection)	Returns a list of stored procedures from the server.
GetTableNames (inherited from TCustomDACConnection)	Provides a list of available tables names.
MonitorMessage (inherited from TCustomDACConnection)	Sends a specified message through the TCustomDASQLMonitor component.
ParamByName	POrovides access to the

	OUT parameters and their values after processing SQL statement.
Ping (inherited from TCustomDACConnection)	Used to check state of connection to the server.
RemoveFromPool (inherited from TCustomDACConnection)	Marks the connection that should not be returned to the pool after disconnect.
RemoveTransaction	Disassociates a specified transaction from the database.
Rollback (inherited from TCustomDACConnection)	Discards all current data changes and ends transaction.
RollbackRetaining	Rolls back all changes of data associated with the default database transaction to the database server and then retains the transaction context.
StartTransaction (inherited from TCustomDACConnection)	Begins a new user transaction.

Events

Name	Description
OnConnectionLost (inherited from TCustomDACConnection)	This event occurs when connection was lost.
OnError (inherited from TCustomDACConnection)	This event occurs when an error has arisen in the connection.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2 Properties

Properties of the **TIBConnection** class.

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

Public

Name	Description
ConnectDialog (inherited from TCustomDAConnection)	Allows to link a TCustomConnectDialog component.
ConnectionString (inherited from TCustomDAConnection)	Used to specify the connection information, such as: UserName, Password, Server, etc.
ConvertEOL (inherited from TCustomDAConnection)	Allows customizing line breaks in string fields and parameters.
DatabaseInfo	Used to get information about the connected database.
DBSQLDialect	Shows the database SQL dialect.
Handle	Used to make calls directly to the InterBase API.
InTransaction (inherited from TCustomDAConnection)	Indicates whether the transaction is active.
LastError	Used to get the error code which resulted from the previous call to the InterBase server.
LoginPrompt (inherited from TCustomDAConnection)	Specifies whether a login dialog appears immediately before opening a new connection.
Pooling (inherited from TCustomDAConnection)	Enables or disables using connection pool.
PoolingOptions (inherited from TCustomDAConnection)	Specifies the behaviour of connection pool.
SQL	Used to execute any SQL statement.
TransactionCount	Used to return the number of transactions currently associated with this TIBCCConnection component.
Transactions	Used to specify a transaction for the given

	index.
--	--------

Published

Name	Description
AutoCommit	Used to permit or prevent permanent updates, insertions, and deletions of data associated with the current transaction against the database server.
ClientLibrary	Used to set or get the client library location.
Connected	Used to establish a database connection.
ConnectPrompt	Used to provide login support.
Database	Used to set the name of the database to associate with TIBCConnection component.
Debug	Used to display SQL statements being executed with their parameter values and data types.
DefaultTransaction	Used to access default database connection transaction.
Options	Used to specify the behaviour of the TIBCConnection object.
Params	Used to specify the connection parameters.
Password	Used to specify the password for a connection.
Port	Used to specify the port number for connections to the server.
Server	Used to supply the server name to handle server's request for a login.
SQLDialect	Used to set or return SQL Dialect used by InterBase client.

SSLOptions	Used to set up the SSL options.
Username	Used to provide a user name.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.1 AutoCommit Property

Used to permit or prevent permanent updates, insertions, and deletions of data associated with the current transaction against the database server.

Class

[TIBConnection](#)

Syntax

```
property AutoCommit: boolean;
```

Remarks

Use the AutoCommit property to permit or prevent permanent updates, insertions, and deletions of data associated with the current transaction against the database server without explicit calls to Commit or Rollback methods.

Set AutoCommit to True to permit implicit call to Commit method after every database access.

AutoCommit property in TIBConnection has higher precedence over the same properties in dataset components

The default value is True.

Note: The AutoCommit property in TIBConnection globally specifies whether all queries to modify database are implicitly committed or not. Components which descend from

[TCustomDADataset](#) and [TCustomDASQL](#) classes inherit their AutoCommit properties. This allows them to selectively specify their implicit transaction committing behavior after each data modifying access.

See Also

- [TCustomDAConnection.Commit](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.2 ClientLibrary Property

Used to set or get the client library location.

Class

[TIBConnection](#)

Syntax

```
property ClientLibrary: string;
```

Remarks

Use the ClientLibrary property to set or get the client library location.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.3 Connected Property

Used to establish a database connection.

Class

[TIBConnection](#)

Syntax

```
property Connected stored IsConnectedStored;
```

Remarks

Indicates whether the database connection is active. Set this property to True to establish a database connection. Setting this property to False will close a connection.

See Also

- [TCustomDACConnection.Connect](#)
- [TCustomDACConnection.Disconnect](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.4 ConnectPrompt Property

Used to provide login support.

Class

[TIBConnection](#)

Syntax

```
property ConnectPrompt: boolean stored False default True;
```

Remarks

Set ConnectPrompt to True to provide login support when establishing a connection. When ConnectPrompt is True, a dialog appears to prompt a user for a name and a password.

When ConnectPrompt is False, an application must supply user name and password values programmatically.

Warning: Storing hard-coded user name and password entries as property values or in code for an OnLogin event handler can compromise the server security.

See Also

- [Password](#)
- [Server](#)
- [Username](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.5 Database Property

Used to set the name of the database to associate with TIBCCConnection component.

Class

[TIBCCConnection](#)

Syntax

```
property Database: string;
```

Remarks

Use the Database property to set the name of the database to associate with TIBCCConnection component. For local InterBase databases it is a filename. For remote databases use following syntax.

To connect to an InterBase database on a remote server using TCP/IP the syntax is <server_name>:<filename>.

To connect to an InterBase database on a remote server using NetBEUI, the syntax is: \<server_name>\<filename>.

To connect to an InterBase database on a remote server using SPX, the syntax is: <server_name>@<filename>.

Note: You can set connection parameters not only by Database property. You can use Server property to set server name and Options property to set connection protocol. If Database property contains local filename and server property is empty, it will be a local connection. If Server property is not empty, Server and Options properties will be used for the connection. If both Database and Server property contain server name and they are the same, server name and connection protocol will be taken from the Database property. If they are not the same, the server name and connection protocol will be taken from Server and Options properties and entire string of Database property will be processed as a database name without server name. In that case you may encounter errors.

See Also

- [Options](#)
- [Server](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.6 DatabaseInfo Property

Used to get information about the connected database.

Class

[TIBConnection](#)

Syntax

```
property DatabaseInfo: TGDSDatabaseInfo;
```

Remarks

Use the DatabaseInfo property to get information about the connected database. You should explicitly add the T:Devart.IbDac.Units.IBCClasses unit to the 'uses' list to use this property.

See Also

- [TGDSDatabaseInfo](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.7 DBSQLDialect Property

Shows the database SQL dialect.

Class

[TIBConnection](#)

Syntax

```
property DBSQLDialect: Integer;
```

Remarks

The DBSQLDialect property is used to show the database SQL dialect.

See Also

- [SQLDialect](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.8 Debug Property

Used to display SQL statements being executed with their parameter values and data types.

Class

[TIBConnection](#)

Syntax

```
property Debug: boolean;
```

Remarks

Set the Debug property to True to display SQL statements being executed with their parameter values and data types.

Note: To use this property you should explicitly include unit IBDACVcl (IBDACClx under Linux) to your project.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.9 DefaultTransaction Property

Used to access default database connection transaction.

Class

[TIBConnection](#)

Syntax

```
property DefaultTransaction: TIBTransaction;
```

Remarks

Use the DefaultTransaction property to access default database connection transaction. By default this is internal connection transaction. You can set it to external transaction

component. To restore internal transaction set this property to nil.

See Also

- [Transactions](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.10 Handle Property

Used to make calls directly to the InterBase API.

Class

[TIBConnection](#)

Syntax

```
property Handle: TISC_DB_HANDLE;
```

Remarks

Use the Handle property to make calls directly to the InterBase API. Many of the InterBase API functions require a database handle as an argument.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.11 LastError Property

Used to get the error code which resulted from the previous call to the InterBase server.

Class

[TIBConnection](#)

Syntax

```
property LastError: integer;
```

Remarks

Use the LastError property to get the error code which resulted from the previous call to the

InterBase server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.12 Options Property

Used to specify the behaviour of the TIBConnection object.

Class

[TIBConnection](#)

Syntax

```
property Options: TIBConnectionOptions;
```

Remarks

Set properties of Options to specify the behaviour of a TIBConnection object.

Descriptions of all options are in the table below.

Option Name	Description
CharLength	Used to specify the size in bytes of a single character.
Charset	Used to set character set that IBDAC uses to read and write character data.
EnableBCD	Used to enable currency type.
EnableFMTBCD	Used to enable using FMTBCD instead of float for large integer numbers to keep precision.
EnableMemos	Used to enable creating TMemoField and TWideMemoField for BLOB fields with subtype 1.
IPVersion	Used to specify Internet Protocol Version.
NoDBTriggers	Used to disable all database triggers.
Protocol	Used to specify the Network protocol of connection with InterBase server.
Role	Used to specify the InterBase connection role.
TrustedAuthentication	Used to apply Windows "Trusted User" security for authenticating Firebird users.
UseUnicode	Used to enable or disable Unicode

	support.
--	----------

See Also

- [TCustomDACConnection.Options](#)
- [Unicode Character Data](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.13 Params Property

Used to specify the connection parameters.

Class

[TIBConnection](#)

Syntax

```
property Params: TStrings;
```

Remarks

Use the Params property to specify the connection parameters, such as the username, password, role, and more. For example: user_name=sysdba password=masterkey sql_role_name=role1 lc_ctype=WIN1252

Note that TIBConnection also has the Username and Password properties, therefore you should use either Username and Password or Params, but not both.

If you are using the Params property to create a new database with the CreateDatabase function, you should set the following parameters: USER 'SYSDBA' PASSWORD 'masterkey' PAGE_SIZE 4096

You can enable compression of data over the wire at global or individual database level by passing WireCompression=True to Params. WireCompression is disabled by default. Note that zlib1.dll should be placed in the same location as fbclient.dll.

Example

Runtime examples:

```
Params.Add('user_name=sysdba');  
Params.Add('password=masterkey');  
Params.Add('sql_role_name=role1');  
----  
Params.Add('USER ''SYSDBA''');  
Params.Add('PASSWORD ''masterkey''');  
Params.Add('PAGE_SIZE 4096');
```

See Also

- [CreateDatabase](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.14 Password Property

Used to specify the password for a connection.

Class

[TIBCCConnection](#)

Syntax

```
property Password: string;
```

Remarks

Use the Password property to specify a password for connection.

When property is being changed TIBCCConnection calls the Disconnect method.

See Also

- [Username](#)
- [Server](#)
- [TCustomDACConnection.Connect](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.15 Port Property

Used to specify the port number for connections to the server.

Class

[TIBConnection](#)

Syntax

```
property Port: string;
```

Remarks

Use the Port property to specify the port number for connections to InterBase and Firebird.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.16 Server Property

Used to supply the server name to handle server's request for a login.

Class

[TIBConnection](#)

Syntax

```
property Server: string;
```

Remarks

Use the Server property to supply the server name to handle server's request for a login.

When property is being changed TIBConnection calls the Disconnect method.

See Also

- [Username](#)
- [Password](#)
- [TCustomDACConnection.Connect](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.11.1.6.2.17 SQL Property

Used to execute any SQL statement.

Class

[TIBConnection](#)

Syntax

```
property SQL: TIBCSQL;
```

Remarks

You can use the embedded TIBCSQL object to execute any SQL statement.

See Also

- [TIBCSQL](#)
- [TCustomDACConnection.ExecSQL](#)
- [TCustomDACConnection.ExecSQLEx](#)
- [ParamByName](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.18 SQLDialect Property

Used to set or return SQL Dialect used by InterBase client.

Class

[TIBConnection](#)

Syntax

```
property SQLDialect: Integer default 3;
```

Remarks

Use the SQLDialect property to set or return SQL Dialect used by InterBase client. The

SQLDialect property cannot be set to a value greater than the database SQL dialect when the connection is active. If the connection is inactive, the SQLDialect property will be downgraded to match the database SQL dialect.

See Also

- [DBSQLDialect](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.19 SSLOptions Property

Used to set up the SSL options.

Class

[TIBConnection](#)

Syntax

```
property SSLOptions: TIBCSSLConnectionOptions;
```

Remarks

Use the SSLOptions property to set up the SSL options.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.20 TransactionCount Property

Used to return the number of transactions currently associated with this TIBConnection component.

Class

[TIBConnection](#)

Syntax

```
property TransactionCount: integer;
```

Remarks

Use the TransactionCount property to return the number of transactions currently associated with this TIBCCConnection component.

See Also

- [Transactions](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.21 Transactions Property(Indexer)

Used to specify a transaction for the given index.

Class

[TIBCCConnection](#)

Syntax

```
property Transactions[Index: Integer]: TIBCTransaction;
```

Parameters

Index

Holds the specified index.

Remarks

Use the Transactions property to specify a transaction for the given index.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.2.22 Username Property

Used to provide a user name.

Class

[TIBCCConnection](#)

Syntax

```
property Username: string;
```

Remarks

Use the Username property to supply the user name to handle server's request for a login.

When property is being changed TIBConnection calls Disconnect method.

See Also

- [Password](#)
- [Server](#)
- [TCustomDAConnection.Connect](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.3 Methods

Methods of the **TIBConnection** class.

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

Public

Name	Description
AddTransaction	Associates a TIBCTransaction component with the database component.
ApplyUpdates (inherited from TCustomDAConnection)	Overloaded. Applies changes in datasets.
AssignConnect	Shares database connection between the TIBConnection components.
Commit (inherited from TCustomDAConnection)	Commits current transaction.
CommitRetaining	Stores to the database server all changes of data associated with the default database transaction permanently and then retains the transaction context.

Connect (inherited from TCustomDACConnection)	Establishes a connection to the server.
CreateDatabase	Creates an InterBase database.
CreateSQL (inherited from TCustomDACConnection)	Creates a component for queries execution.
Disconnect (inherited from TCustomDACConnection)	Performs disconnect.
DropDatabase	Used to drop database and remove database file from server.
ExecProc (inherited from TCustomDACConnection)	Allows to execute stored procedure or function providing its name and parameters.
ExecProcEx (inherited from TCustomDACConnection)	Allows to execute a stored procedure or function.
ExecSQL (inherited from TCustomDACConnection)	Executes a SQL statement with parameters.
ExecSQLEx (inherited from TCustomDACConnection)	Executes any SQL statement outside the TQuery or TSQL components.
FindDefaultTransaction	Returns the default transaction for the database.
GetDatabaseNames (inherited from TCustomDACConnection)	Returns a database list from the server.
GetKeyFieldNames (inherited from TCustomDACConnection)	Provides a list of available key field names.
GetStoredProcNames (inherited from TCustomDACConnection)	Returns a list of stored procedures from the server.
GetTableNames (inherited from TCustomDACConnection)	Provides a list of available tables names.
MonitorMessage (inherited from TCustomDACConnection)	Sends a specified message through the TCustomDASQLMonitor component.
ParamByName	POvides access to the OUT parameters and their values after processing SQL

	statement.
Ping (inherited from TCustomDACConnection)	Used to check state of connection to the server.
RemoveFromPool (inherited from TCustomDACConnection)	Marks the connection that should not be returned to the pool after disconnect.
RemoveTransaction	Disassociates a specified transaction from the database.
Rollback (inherited from TCustomDACConnection)	Discards all current data changes and ends transaction.
RollbackRetaining	Rolls back all changes of data associated with the default database transaction to the database server and then retains the transaction context.
StartTransaction (inherited from TCustomDACConnection)	Begins a new user transaction.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.3.1 AddTransaction Method

Associates a TIBCTransaction component with the database component.

Class

[TIBConnection](#)

Syntax

```
function AddTransaction(TR: TIBCTransaction): Integer;
```

Parameters

TR

Holds the transaction that is being added.

Return Value

the index of the associated transaction in the transaction list.

Remarks

Use the AddTransaction method to associate a TIBCTransaction component with the database component. Returns the index of the associated transaction in the transaction list.

See Also

- [Transactions](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.3.2 AssignConnect Method

Shares database connection between the TIBConnection components.

Class

[TIBConnection](#)

Syntax

```
procedure AssignConnect(Source: TIBConnection);
```

Parameters

Source

Holds the source connection.

Remarks

Call the AssignConnect method to share database connection between the TIBConnection components.

AssignConnect assumes that Source parameter points to a preconnected database component and sets for this instance of TIBConnection Connected property to True. Note that AssignConnect doesn't make any references to the Source database. So before disconnecting parent TIBConnection call AssignConnect(Nil) or Disconnect method for all assigned databases.

See Also

- [TCustomDACConnection.Connect](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.3.3 CommitRetaining Method

Stores to the database server all changes of data associated with the default database transaction permanently and then retains the transaction context.

Class

[TIBConnection](#)

Syntax

```
procedure CommitRetaining;
```

Remarks

Call the CommitRetaining method to store to the database server all changes of data associated with the default database transaction permanently and then retain the transaction context.

See Also

- [DefaultTransaction](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.3.4 CreateDatabase Method

Creates an InterBase database.

Class

[TIBConnection](#)

Syntax

```
procedure CreateDatabase;
```

Remarks

Call the CreateDatabase method to create an InterBase database using [Params](#) as the rest of the CREATE DATABASE statement.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.3.5 DropDatabase Method

Used to drop database and remove database file from server.

Class

[TIBConnection](#)

Syntax

```
procedure DropDatabase;
```

Remarks

Call the DropDatabase method to drop database and remove database file from server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.3.6 FindDefaultTransaction Method

Returns the default transaction for the database.

Class

[TIBConnection](#)

Syntax

```
function FindDefaultTransaction: TIBTransaction;
```

Return Value

the default transaction for the database.

Remarks

Call the FindDefaultTransaction method to return the default transaction for the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.3.7 ParamByName Method

POrovides access to the OUT parameters and their values after processing SQL statement.

Class

[TIBConnection](#)

Syntax

```
function ParamByName(const Name: string): TIBParam;
```

Parameters

Name

Holds the parameter name.

Return Value

the corresponding parameter.

Remarks

Call the ParamByName method to get access to the OUT parameters and their values after processing SQL statement with ExecSQL or stored procedure with ExecProc. Name should be equal to the parameter name as it occurred in SQL statement.

Implicitly calls the [TIBCSQL.ParamByName](#) function of [TIBCSQL](#).

See Also

- [SQL](#)
- [TCustomDACConnection.ExecSQL](#)
- [TCustomDACConnection.ExecSQLEx](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.3.8 RemoveTransaction Method

Disassociates a specified transaction from the database.

Class

[TIBConnection](#)

Syntax

```
procedure RemoveTransaction(TR: TIBTransaction);
```

Parameters

TR

Remarks

Call the RemoveTransaction method to disassociate a specified transaction from the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.6.3.9 RollbackRetaining Method

Rolls back all changes of data associated with the default database transaction to the database server and then retains the transaction context.

Class

[TIBConnection](#)

Syntax

```
procedure RollbackRetaining;
```

Remarks

Call the RollbackRetaining method to roll back all changes of data associated with the default database transaction to the database server and then retain the transaction context.

See Also

- [DefaultTransaction](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7 TIBConnectionOptions Class

This class allows setting up the behaviour of the TIBConnection class.

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

Unit

[IBC](#)

Syntax

```
TIBConnectionOptions = class(TDACConnectionOptions);
```

Inheritance Hierarchy

[TDACConnectionOptions](#)

TIBConnectionOptions

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7.1 Members

[TIBConnectionOptions](#) class overview.

Properties

Name	Description
AllowImplicitConnect (inherited from TDACConnectionOptions)	Specifies whether to allow or not implicit connection opening.
CharLength	Used to specify the size in bytes of a single character.
Charset	Used to set character set that IBDAC uses to read and write character data.
DefaultSortType (inherited from TDACConnectionOptions)	Used to determine the default type of local sorting for string fields. It is used

	when a sort type is not specified explicitly after the field name in the TMemDataSet.IndexFieldNames property of a dataset.
DisconnectedMode (inherited from TDACConnectionOptions)	Used to open a connection only when needed for performing a server call and closes after performing the operation.
EnableBCD	Used to enable currency type.
EnableFMTBCD	Used to enable using FMTBCD instead of float for large integer numbers to keep precision.
EnableMemos	Used to enable creating TMemoField and TWideMemoField for BLOB fields with subtype 1.
IPVersion	Used to specify Internet Protocol Version.
KeepDesignConnected (inherited from TDACConnectionOptions)	Used to prevent an application from establishing a connection at the time of startup.
LocalFailover (inherited from TDACConnectionOptions)	If True, the TCustomDAConnection.OnConnectionLost event occurs and a failover operation can be performed after connection breaks.
NoDBTriggers	Used to disable all database triggers.
Protocol	Used to specify the Network protocol of connection with InterBase server.
Role	Used to specify the InterBase connection role.
TrustedAuthentication	Used to apply Windows "Trusted User" security for authenticating Firebird users.
UseUnicode	Used to enable or disable

Unicode support.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.11.1.7.2 Properties

Properties of the **TIBConnectionOptions** class.

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

Public

Name	Description
DefaultSortType (inherited from TDACConnectionOptions)	Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the TMemDataSet.IndexFieldNames property of a dataset.
DisconnectedMode (inherited from TDACConnectionOptions)	Used to open a connection only when needed for performing a server call and closes after performing the operation.
KeepDesignConnected (inherited from TDACConnectionOptions)	Used to prevent an application from establishing a connection at the time of startup.
LocalFailover (inherited from TDACConnectionOptions)	If True, the TCustomDAConnection.OnConnectionLost event occurs and a failover operation can be performed after connection breaks.

Published

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

AllowImplicitConnect (inherited from TDACConnectionOptions)	Specifies whether to allow or not implicit connection opening.
CharLength	Used to specify the size in bytes of a single character.
Charset	Used to set character set that IBDAC uses to read and write character data.
EnableBCD	Used to enable currency type.
EnableFMTBCD	Used to enable using FMTBCD instead of float for large integer numbers to keep precision.
EnableMemos	Used to enable creating TMemoField and TWideMemoField for BLOB fields with subtype 1.
IPVersion	Used to specify Internet Protocol Version.
NoDBTriggers	Used to disable all database triggers.
Protocol	Used to specify the Network protocol of connection with InterBase server.
Role	Used to specify the InterBase connection role.
TrustedAuthentication	Used to apply Windows "Trusted User" security for authenticating Firebird users.
UseUnicode	Used to enable or disable Unicode support.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7.2.1 CharLength Property

Used to specify the size in bytes of a single character.

Class

[TIBConnectionOptions](#)

Syntax

```
property CharLength: TIBCharLength default 0;
```

Remarks

Use the CharLength property to specify the size in bytes of a single character. Set this option with the number in range [0..6] to reflect InterBase support for the national languages. Setting CharLength to zero will instruct TIBConnection to interrogate InterBase server for the actual character length.

The default value is 1.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7.2.2 Charset Property

Used to set character set that IBDAC uses to read and write character data.

Class

[TIBConnectionOptions](#)

Syntax

```
property Charset: string;
```

Remarks

Use the Charset property to set character set that IBDAC uses to read and write character data.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7.2.3 EnableBCD Property

Used to enable currency type.

Class

[TIBConnectionOptions](#)

Syntax

```
property EnableBCD: boolean;
```

Remarks

Use the EnableBCD property to enable currency type.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7.2.4 EnableFMTBCD Property

Used to enable using FMTBCD instead of float for large integer numbers to keep precision.

Class

[TIBConnectionOptions](#)

Syntax

```
property EnableFMTBCD: boolean;
```

Remarks

Use the EnableFMTBCD property to enable using FMTBCD instead of float for large integer numbers to keep precision.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7.2.5 EnableMemos Property

Used to enable creating TMemoField and TWideMemoField for BLOB fields with subtype 1.

Class

[TIBConnectionOptions](#)

Syntax

```
property EnableMemos: boolean default False;
```

Remarks

Use the EnableMemos property to enable creating TMemoField and TWideMemoField for BLOB fields with subtype 1.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7.2.6 IPVersion Property

Used to specify Internet Protocol Version.

Class

[TIBConnectionOptions](#)

Syntax

```
property IPVersion: TIPVersion default DefValIPVer;
```

Remarks

Use the IPVersion property to specify Internet Protocol Version.

Supported values:

- ivIPBoth (default) - specifies that either Internet Protocol Version 6 (IPv6) or Version 4 (IPv4) will be used;
- ivIPv4 - specifies that Internet Protocol Version 4 (IPv4) will be used;
- ivIPv6 - specifies that Internet Protocol Version 6 (IPv6) will be used.

Note : Internet Protocol Version support has been added in Firebird 3. To use the IPVersion option, your client library version must be version 3 or higher.

When the TIPVersion property is set to ivIPBoth, a connection attempt will be made via IPv6 if it is enabled on the operating system. If the connection attempt fails, a new connection

attempt will be made via IPv4.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7.2.7 NoDBTriggers Property

Used to disable all database triggers.

Class

[TIBConnectionOptions](#)

Syntax

```
property NoDBTriggers: boolean default False;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7.2.8 Protocol Property

Used to specify the Network protocol of connection with InterBase server.

Class

[TIBConnectionOptions](#)

Syntax

```
property Protocol: TIBProtocol default DefValProtocol;
```

Remarks

Use the Protocol property to specify the network protocol of connection with InterBase server. The default value is TCP.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7.2.9 Role Property

Used to specify the InterBase connection role.

Class

[TIBConnectionOptions](#)

Syntax

```
property Role: string;
```

Remarks

Use the Role property to specify the InterBase connection role.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7.2.10 TrustedAuthentication Property

Used to apply Windows "Trusted User" security for authenticating Firebird users.

Class

[TIBConnectionOptions](#)

Syntax

```
property TrustedAuthentication: boolean default False;
```

Remarks

Use the TrustedAuthentication property to apply Windows "Trusted User" security for authenticating Firebird users on a Windows host. When the option is set to True, the Firebird security database is ignored during establishing a connection, and only Windows authentication is used.

The default value is False. More detailed information about this authentication mode is available at <http://firebirdsql.org/rlsnotes/rlsnotes210.html#rnf210-wintrusted> .

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.7.2.11 UseUnicode Property

Used to enable or disable Unicode support.

Class

[TIBConnectionOptions](#)

Syntax

```
property UseUnicode: boolean default DefValUseUnicode;
```

Remarks

Use the UseUnicode property to enable or disable Unicode support. Affects on character data fetched from the server. When set to True all character data is stored as WideStrings, and TStringField is replaced with TWideStringField.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8 TIBDataSetOptions Class

This class allows setting up the behaviour of the TIBDataSet class.

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

Unit

[IBC](#)

Syntax

```
TIBDataSetOptions = class(TDADatasetOptions);
```

Inheritance Hierarchy

[TDADatasetOptions](#)

TIBDataSetOptions

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.1 Members

[TIBCDataSetOptions](#) class overview.

Properties

Name	Description
AutoClose	Used to for CustomIBCDataSet to close cursor after fetching all rows.
AutoPrepare (inherited from TDADatasetOptions)	Used to execute automatic TCustomDADataset.Prepare on the query execution.
BooleanDomainFields	Used to create TBooleanField for fields that have domain of the integer data type, and the domain name contains 'BOOLEAN'.
CacheArrays	Used to allocate local memory buffer for a copy of the array.
CacheBlobs	Used to allocate local memory buffer for a copy of the BLOB.
CacheCalcFields (inherited from TDADatasetOptions)	Used to enable caching of the TField.Calculated and TField.Lookup fields.
ComplexArrayFields	Used to store array fields as TIBCArrayField objects.
CompressBlobMode (inherited from TDADatasetOptions)	Used to store values of the BLOB fields in compressed form.
DefaultValues	Used for TCustomIBCDataSet to fill the DefaultExpression property of TField objects by the appropriate value.
DeferredArrayRead	Used for fetching all InterBase array values when they are explicitly requested.
DeferredBlobRead	Used for fetching all InterBase BLOB values when they are explicitly requested.

DescribeParams	Used to specify whether to query TBCParam properties from the server when executing the TCustomDADataset.Prepare method.
DetailDelay (inherited from TDADatasetOptions)	Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.
ExtendedFieldsInfo	Used to perform an additional query to get information about the returned fields and the tables they belong to.
FieldsAsString	Used to treat all non-BLOB fields as being of string datatype.
FieldsOrigin (inherited from TDADatasetOptions)	Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.
FlatBuffers (inherited from TDADatasetOptions)	Used to control how a dataset treats data of the ftString and ftVarBytes fields.
FullRefresh	Used to refresh fields from all tables of the query.
InsertAllSetFields (inherited from TDADatasetOptions)	Used to include all set dataset fields in the generated INSERT statement
LocalMasterDetail (inherited from TDADatasetOptions)	Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.
LongStrings (inherited from TDADatasetOptions)	Used to represent string fields with the length that is greater than 255 as TStringField.

MasterFieldsNullable (inherited from TDADatasetOptions)	Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this option is enabled, the performance can get worse).
NumberRange (inherited from TDADatasetOptions)	Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.
PrepareUpdateSQL	Used to automatically prepare update queries before execution.
QueryRecCount (inherited from TDADatasetOptions)	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.
QueryRowsAffected	Used to increase the performance of update operations.
QuoteNames (inherited from TDADatasetOptions)	Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.
RemoveOnRefresh (inherited from TDADatasetOptions)	Used for a dataset to locally remove a record that can not be found on the server.
RequiredFields (inherited from TDADatasetOptions)	Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.
ReturnParams (inherited from TDADatasetOptions)	Used to return the new value of fields to dataset after insert or update.
SetDomainNames	Used to retrieve the DOMAIN name for a field.

SetEmptyStrToNull	Force replace of empty strings with NULL values in data. The default value is False.
SetFieldsReadOnly (inherited from TDADatasetOptions)	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.
StreamedBlobs	Used to handle and save BLOBs as streamed BLOBs.
StrictUpdate	Used for TCustomIBCDataSet to raise an exception when the number of the updated or deleted records is not equal 1.
TrimFixedChar (inherited from TDADatasetOptions)	Specifies whether to discard all trailing spaces in the string fields of a dataset.
UpdateAllFields (inherited from TDADatasetOptions)	Used to include all dataset fields in the generated UPDATE and INSERT statements.
UpdateBatchSize (inherited from TDADatasetOptions)	Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2 Properties

Properties of the **TIBCDataSetOptions** class.

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

Public

Name	Description
AutoPrepare (inherited from TDADatasetOptions)	Used to execute automatic TCustomDADataset.Prepare on the query execution.
CacheCalcFields (inherited from TDADatasetOptions)	Used to enable caching of the TField.Calculated and TField.Lookup fields.
CompressBlobMode (inherited from TDADatasetOptions)	Used to store values of the BLOB fields in compressed form.
DetailDelay (inherited from TDADatasetOptions)	Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.
FieldsOrigin (inherited from TDADatasetOptions)	Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.
FlatBuffers (inherited from TDADatasetOptions)	Used to control how a dataset treats data of the ftString and ftVarBytes fields.
InsertAllSetFields (inherited from TDADatasetOptions)	Used to include all set dataset fields in the generated INSERT statement
LocalMasterDetail (inherited from TDADatasetOptions)	Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.
LongStrings (inherited from TDADatasetOptions)	Used to represent string fields with the length that is greater than 255 as TStringField.
MasterFieldsNullable (inherited from TDADatasetOptions)	Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this

	option is enabled, the performance can get worse).
NumberRange (inherited from TDADatasetOptions)	Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.
QueryRecCount (inherited from TDADatasetOptions)	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 (inherited from TDADatasetOptions)	Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.
RemoveOnRefresh (inherited from TDADatasetOptions)	Used for a dataset to locally remove a record that can not be found on the server.
RequiredFields (inherited from TDADatasetOptions)	Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.
ReturnParams (inherited from TDADatasetOptions)	Used to return the new value of fields to dataset after insert or update.
SetFieldsReadOnly (inherited from TDADatasetOptions)	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.
TrimFixedChar (inherited from TDADatasetOptions)	Specifies whether to discard all trailing spaces in the string fields of a dataset.
UpdateAllFields (inherited from TDADatasetOptions)	Used to include all dataset fields in the generated UPDATE and INSERT statements.

UpdateBatchSize (inherited from TDADatasetOptions)	Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.
---	---

Published

Name	Description
AutoClose	Used to for CustomIBCDataset to close cursor after fetching all rows.
BooleanDomainFields	Used to create TBooleanField for fields that have domain of the integer data type, and the domain name contains 'BOOLEAN'.
CacheArrays	Used to allocate local memory buffer for a copy of the array.
CacheBlobs	Used to allocate local memory buffer for a copy of the BLOB.
ComplexArrayFields	Used to store array fields as TIBCArrayField objects.
DefaultValues	Used for TCustomIBCDataset to fill the DefaultExpression property of TField objects by the appropriate value.
DeferredArrayRead	Used for fetching all InterBase array values when they are explicitly requested.
DeferredBlobRead	Used for fetching all InterBase BLOB values when they are explicitly requested.
DescribeParams	Used to specify whether to query TBCParam properties from the server when executing the TCustomDADataset.Prepare method.

ExtendedFieldsInfo	Used to perform an additional query to get information about the returned fields and the tables they belong to.
FieldsAsString	Used to treat all non-BLOB fields as being of string datatype.
FullRefresh	Used to refresh fields from all tables of the query.
PrepareUpdateSQL	Used to automatically prepare update queries before execution.
QueryRowsAffected	Used to increase the performance of update operations.
SetDomainNames	Used to retrieve the DOMAIN name for a field.
SetEmptyStrToNull	Force replace of empty strings with NULL values in data. The default value is False.
StreamedBlobs	Used to handle and save BLOBs as streamed BLOBs.
StrictUpdate	Used for TCustomIBCDataSet to raise an exception when the number of the updated or deleted records is not equal 1.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.1 AutoClose Property

Used to for CustomIBCDataset to close cursor after fetching all rows.

Class

[TIBCDatasetOptions](#)

Syntax

```
property AutoClose: boolean default False;
```

Remarks

Use the AutoClose property for CustomIBCDataset to close cursor after fetching all rows.
Allows to reduce the number of opened cursors on the server.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.2 BooleanDomainFields Property

Used to create TBooleanField for fields that have domain of the integer data type, and the domain name contains 'BOOLEAN'.

Class

[TIBCDatasetOptions](#)

Syntax

```
property BooleanDomainFields: boolean default False;
```

Remarks

If the BooleanDomainFields property is set to True, TBooleanField objects are created for fields that have domain of the integer data type, and the domain name contains 'BOOLEAN'.

Note: This option has no effect when [SetDomainNames](#) is set to False.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.3 CacheArrays Property

Used to allocate local memory buffer for a copy of the array.

Class

[TIBCDatasetOptions](#)

Syntax

```
property CacheArrays: boolean default True;
```

Remarks

If the CacheArray property is set to True, local memory buffer is allocated for a copy of the array.

Note: This option has no effect when [DeferredArrayRead](#) is set to False because all BLOB values are fetched to the dataset in that case.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.4 CacheBlobs Property

Used to allocate local memory buffer for a copy of the BLOB.

Class

[TIBCDatasetOptions](#)

Syntax

```
property CacheBlobs: boolean default True;
```

Remarks

If the CacheBlobs property is set to True, local memory buffer is allocated for a copy of the BLOB.

Note: The CacheBlobs option controls the way streamed BLOB objects are handled. If False, application can access streamed BLOB values on the server without caching BLOBs on the client - only the requested portions of data are fetched. Setting CacheBlobs to False may reduce network traffic since only the required data is fetched, and reduce memory

consumption on the client because the returned record sets do not hold contents of BLOB fields.

This feature is only available for streamed BLOBs when [StreamedBlobs](#) is set to True.

This option has no effect if [DeferredBlobRead](#) is set to False because all BLOB values are fetched to the dataset in that case.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.5 ComplexArrayFields Property

Used to store array fields as TIBCArrayField objects.

Class

[TIBCDataSetOptions](#)

Syntax

```
property ComplexArrayFields: boolean default True;
```

Remarks

If the ComplexArrayFields property is set to False, any array field is stored as one TIBCArrayField object. If true and ObjectView is true, array items are stored hierarchically. If true and ObjectView is false, all array items are stored as sibling fields.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.6 DefaultValues Property

Used for TCustomIBCDataset to fill the DefaultExpression property of TField objects by the appropriate value.

Class

[TIBCDataSetOptions](#)

Syntax

```
property DefaultValues: boolean;
```

Remarks

If the `DefaultValues` property is set to `True`, `TCustomIBCDataset` fills the `DefaultExpression` property of `TField` objects by the appropriate value. Note that computed BLR fields are not detected and set to read-only if `DefaultValues` set to `false`.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.7 DeferredArrayRead Property

Used for fetching all InterBase array values when they are explicitly requested.

Class

[TIBCDatasetOptions](#)

Syntax

```
property DeferredArrayRead: boolean default True;
```

Remarks

If the `DeferredArrayRead` property is set to `True`, all InterBase array values are only fetched when they are explicitly requested. Otherwise entire record set with any array values is returned when dataset is opened. Whether array values are cached locally to be reused later or not is controlled by the `CacheBlobs` option.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.8 DeferredBlobRead Property

Used for fetching all InterBase BLOB values when they are explicitly requested.

Class

[TIBCDatasetOptions](#)

Syntax

```
property DeferredBlobRead: boolean default False;
```

Remarks

If the `DeferredBlobRead` property is set to `True`, all InterBase BLOB values are only fetched when they are explicitly requested. Otherwise entire record set with any BLOB values is returned when dataset is opened. Whether BLOB values are cached locally to be reused later or not is controlled by `CacheBlobs` option.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.9 DescribeParams Property

Used to specify whether to query [TIBCPParam](#) properties from the server when executing the [TCustomDADataset.Prepare](#) method.

Class

[TIBCDatasetOptions](#)

Syntax

```
property DescribeParams: boolean default False;
```

Remarks

Specifies whether to query [TIBCPParam](#) properties (Name, ParamType, DataType, Size, TableName) from the server when executing the [TCustomDADataset.Prepare](#) method. The default value is `False`.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.10 ExtendedFieldsInfo Property

Used to perform an additional query to get information about the returned fields and the tables they belong to.

Class

[TIBCDatasetOptions](#)

Syntax

```
property ExtendedFieldsInfo: boolean default True;
```

Remarks

Use the ExtendedFieldsInfo property to perform an additional query to get information about the returned fields and the tables they belong to.

If True, an additional query is performed to get information about the returned fields and the tables they belong to. This information includes the NOT NULL attribute of the field, the SEQUENCE linked to the field, and the table name corresponding to the field. The table name is needed to detect fields that belong to the updated table and set the read-only attribute for all other fields returned by the query.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.11 FieldsAsString Property

Used to treat all non-BLOB fields as being of string datatype.

Class

[TIBCDatasetOptions](#)

Syntax

```
property FieldsAsString: boolean default False;
```

Remarks

If the FieldsAsString property is set to True, then all non-BLOB fields are treated as being of string datatype.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.12 FullRefresh Property

Used to refresh fields from all tables of the query.

Class

[TIBCDatasetOptions](#)

Syntax

```
property FullRefresh: boolean;
```

Remarks

Use the FullRefresh property to refresh fields from all tables of the query.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.13 PrepareUpdateSQL Property

Used to automatically prepare update queries before execution.

Class

[TIBDataSetOptions](#)

Syntax

```
property PrepareUpdateSQL: boolean;
```

Remarks

If True, update queries are automatically prepared before executing.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.14 QueryRowsAffected Property

Used to increase the performance of update operations.

Class

[TIBDataSetOptions](#)

Syntax

```
property QueryRowsAffected: boolean default True;
```

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.15 SetDomainNames Property

Used to retrieve the DOMAIN name for a field.

Class

[TIBCDatasetOptions](#)

Syntax

```
property SetDomainNames: boolean default False;
```

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.16 SetEmptyStrToNull Property

Force replace of empty strings with NULL values in data. The default value is False.

Class

[TIBCDatasetOptions](#)

Syntax

```
property SetEmptyStrToNull: boolean;
```

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.17 StreamedBlobs Property

Used to handle and save BLOBs as streamed BLOBs.

Class

[TIBCDatasetOptions](#)

Syntax

```
property streamedBlobs: boolean default False;
```

Remarks

If the StreamedBlobs property is set to True, then all BLOBs are handled and saved as

streamed BLOBs. Otherwise, BLOBs are handled and saved as segmented BLOBs. For more information on BLOBs, see [BLOB Data Types](#).

Setting this option to True allows you to benefit from [CacheBlobs](#).

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.8.2.18 StrictUpdate Property

Used for TCustomIBCDataset to raise an exception when the number of the updated or deleted records is not equal 1.

Class

[TIBCDatasetOptions](#)

Syntax

```
property strictupdate: boolean;
```

Remarks

TCustomIBCDataset raises exception when the number of the updated or deleted records is not equal 1. Setting this option also causes an exception if the RefreshRecord procedure returns more than one record. The exception does not occur when you use non-SQL block. The default value is True. If False, the AffectedRows property is not calculated and becomes equal zero. This can improve performance of query executing, so if you need to execute many data updating statements at once and you don't mind affected rows count, set this property to False.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.9 TIBCDatasource Class

TIBCDatasource provides an interface between an IBDAC dataset components and data-aware controls on a form.

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

Unit

[IBC](#)

Syntax

```
TIBCDDataSource = class(TCRDataSource);
```

Remarks

TIBCDDataSource provides an interface between an IBDAC dataset components and data-aware controls on a form.

TIBCDDataSource inherits its functionality directly from the TDataSource component.

At design-time assign individual data-aware components' DataSource properties from their drop-down listboxes.

Inheritance Hierarchy

[TCRDataSource](#)

TIBCDDataSource

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.9.1 Members

[TIBCDDataSource](#) class overview.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.10 TIBCDbKeyField Class

A class representing the InterBase RDB\$DB_KEY field.

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

Unit

[IBC](#)

Syntax

```
TIBCDbKeyField = class(TBytesField);
```

Remarks

This class represents the InterBase RDB\$DB_KEY field. It was implemented for the text view of DB_KEY values and does not change TBytesField interface.

See Also

- [Updating Data with IBDAC Dataset Components](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.10.1 Members

[TIBCDbKeyField](#) class overview.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.11 TIBCEncryptor Class

The class that performs encrypting and decrypting of data.

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

Unit

[IBC](#)

Syntax

```
TIBCEncryptor = class(TCREncryptor);
```

Inheritance Hierarchy

[TCREncryptor](#)

TIBCEncryptor

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.11.1 Members

[TIBCEncryptor](#) class overview.

Properties

Name	Description
DataHeader (inherited from TCREncryptor)	Specifies whether the additional information is stored with the encrypted data.
EncryptionAlgorithm (inherited from TCREncryptor)	Specifies the algorithm of data encryption.
HashAlgorithm (inherited from TCREncryptor)	Specifies the algorithm of generating hash data.
InvalidHashAction (inherited from TCREncryptor)	Specifies the action to perform on data fetching when hash data is invalid.
Password (inherited from TCREncryptor)	Used to set a password that is used to generate a key for encryption.

Methods

Name	Description
SetKey (inherited from TCREncryptor)	Sets a key, using which data is encrypted.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.12 TIBCMetaData Class

A component for obtaining metainformation about database objects from the server.

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

Unit

[IBC](#)

Syntax

```
TIBCMetaData = class(TDAMetaData);
```

Remarks

The TIBCMetaData component is used to obtain metainformation from the server about objects in the database, such as tables, table columns, stored procedures, etc.

Inheritance Hierarchy

[TMemDataSet](#)

[TDAMetaData](#)

TIBCMetaData

See Also

- [TCustomDADDataSet.Debug](#)
- [TCustomDASQL.Debug](#)
- [DBMonitor](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.12.1 Members

[TIBCMetaData](#) class overview.

Properties

Name	Description
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Connection (inherited from TDAMetaData)	Used to specify a connection object to use to connect to a data store.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
MetaDataKind (inherited from TDAMetaData)	Used to specify which kind of metainformation to show.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
Restrictions (inherited from TDAMetaData)	Used to provide one or more conditions restricting the list of objects to be described.
Transaction	Used to determine the transaction under which queries to the server are executed.
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
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.

EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
GetBlob (inherited from TMemDataSet)	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
GetMetaDataKinds (inherited from TDAMetaData)	Used to get values acceptable in the MetaDataKind property.
GetRestrictions (inherited from TDAMetaData)	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.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.

SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
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
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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.12.2 Properties

Properties of the **TIBCMetaData** class.

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

Public

Name	Description
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Connection (inherited from TDAMetaData)	Used to specify a connection object to use to connect to a data store.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
MetaDataKind (inherited from TDAMetaData)	Used to specify which kind of metainformation to show.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
Restrictions (inherited from TDAMetaData)	Used to provide one or more conditions restricting the list of objects to be described.
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.

Published

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

[Transaction](#)

Used to determine the transaction under which queries to the server are executed.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.12.2.1 Transaction Property

Used to determine the transaction under which queries to the server are executed.

Class

[TIBCMetaData](#)

Syntax

```
property Transaction: TIBCTransaction stored IsTransactionStored;
```

Remarks

Use the Transaction property to determine the transaction under which queries to the server are executed.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.13 TIBCPParam Class

A class that is used to set the values of individual parameters passed with queries or stored procedures.

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

Unit

[IBC](#)

Syntax

```
TIBCPParam = class(TDAParam);
```

Remarks

Use the properties of TIBCPParam to set the value of a parameter. Objects that use parameters create TIBCPParam objects to represent these parameters. For example, TIBCPParam objects are used by TIBCSQL, TCustomIBCDataset.

TIBCPParam 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 how the field is displayed, edited, or calculated that are not needed in a TIBCPParam object. Conversely, TIBCPParam includes properties that indicate how the field value is passed as a parameter.

Inheritance Hierarchy

[TDAParam](#)

TIBCPParam

See Also

- [TCustomIBCDataset](#)
- [TIBCSQL](#)
- [TIBCPParams](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.13.1 Members

[TIBCPParam](#) class overview.

Properties

Name	Description
AsArray	Used to specify the value of the parameter when it represents the value of the InterBase array type.

AsBlob (inherited from TDAParam)	Used to set and read the value of the BLOB parameter as string.
AsBlobRef (inherited from TDAParam)	Used to set and read the value of the BLOB parameter as a TBlob object.
AsFloat (inherited from TDAParam)	Used to assign the value for a float field to a parameter.
AsIbBlob	Used to specify the value of the parameter when it represents the value of BLOB type.
AsInteger (inherited from TDAParam)	Used to assign the value for an integer field to the parameter.
AsLargeInt (inherited from TDAParam)	Used to assign the value for a LargeInteger field to the parameter.
AsMemo (inherited from TDAParam)	Used to assign the value for a memo field to the parameter.
AsMemoRef (inherited from TDAParam)	Used to set and read the value of the memo parameter as a TBlob object.
AsSQLTimeStamp (inherited from TDAParam)	Used to specify the value of the parameter when it represents a SQL timestamp field.
AsString (inherited from TDAParam)	Used to assign the string value to the parameter.
AsWideString (inherited from TDAParam)	Used to assign the Unicode string value to the parameter.
DataType (inherited from TDAParam)	Indicates the data type of the parameter.
IsNull (inherited from TDAParam)	Used to indicate whether the value assigned to a parameter is NULL.
ParamType (inherited from TDAParam)	Used to indicate the type of use for a parameter.
Size (inherited from TDAParam)	Specifies the size of a string type parameter.

Value (inherited from TDAParam)	Used to represent the value of the parameter as Variant.
--	--

Methods

Name	Description
AssignField (inherited from TDAParam)	Assigns field name and field value to a param.
AssignFieldValue (inherited from TDAParam)	Assigns the specified field properties and value to a parameter.
LoadFromFile (inherited from TDAParam)	Places the content of a specified file into a TDAParam object.
LoadFromStream (inherited from TDAParam)	Places the content from a stream into a TDAParam object.
SetBlobData	Sets the parameter value from the memory buffer.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.13.2 Properties

Properties of the **TIBCPParam** class.

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

Public

Name	Description
AsArray	Used to specify the value of the parameter when it represents the value of the InterBase array type.
AsBlob (inherited from TDAParam)	Used to set and read the value of the BLOB parameter as string.
AsBlobRef (inherited from TDAParam)	Used to set and read the value of the BLOB parameter as a TBlob object.

AsFloat (inherited from TDAParam)	Used to assign the value for a float field to a parameter.
AslBlob	Used to specify the value of the parameter when it represents the value of BLOB type.
AsInteger (inherited from TDAParam)	Used to assign the value for an integer field to the parameter.
AsLargeInt (inherited from TDAParam)	Used to assign the value for a LargeInteger field to the parameter.
AsMemo (inherited from TDAParam)	Used to assign the value for a memo field to the parameter.
AsMemoRef (inherited from TDAParam)	Used to set and read the value of the memo parameter as a TBlob object.
AsSQLTimeStamp (inherited from TDAParam)	Used to specify the value of the parameter when it represents a SQL timestamp field.
AsString (inherited from TDAParam)	Used to assign the string value to the parameter.
AsWideString (inherited from TDAParam)	Used to assign the Unicode string value to the parameter.
IsNull (inherited from TDAParam)	Used to indicate whether the value assigned to a parameter is NULL.

Published

Name	Description
DataType (inherited from TDAParam)	Indicates the data type of the parameter.
ParamType (inherited from TDAParam)	Used to indicate the type of use for a parameter.
Size (inherited from TDAParam)	Specifies the size of a string type parameter.
Value (inherited from TDAParam)	Used to represent the value of the parameter as Variant.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.13.2.1 AsArray Property

Used to specify the value of the parameter when it represents the value of the InterBase array type.

Class

[TIBCParm](#)

Syntax

```
property AsArray: TIBCArray;
```

Remarks

Use the AsArray property to specify the value of the parameter when it represents the value of the InterBase array type.

Setting AsArray will set the DataType property to ftArray.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.13.2.2 AsIbBlob Property

Used to specify the value of the parameter when it represents the value of BLOB type.

Class

[TIBCParm](#)

Syntax

```
property AsIbBlob: TIBCBlob;
```

Remarks

Use the `AsIbBlob` property to specify the value of the parameter when it represents the value of BLOB type.

Setting `AsIBCBLOB` will set the `DataType` property to `ftIBCBLOB`.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.13.3 Methods

Methods of the **TIBCPParam** class.

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

Public

Name	Description
AssignField (inherited from TDAParam)	Assigns field name and field value to a param.
AssignFieldValue (inherited from TDAParam)	Assigns the specified field properties and value to a parameter.
LoadFromFile (inherited from TDAParam)	Places the content of a specified file into a TDAParam object.
LoadFromStream (inherited from TDAParam)	Places the content from a stream into a TDAParam object.
SetBlobData	Sets the parameter value from the memory buffer.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.13.3.1 SetBlobData Method

Sets the parameter value from the memory buffer.

Class

[TIBCPParam](#)

Syntax

```
procedure SetBlobData(Buffer: IntPtr; Size: integer);
```

Parameters

Buffer

Holds the pointer to the buffer with data.

Size

Holds the buffer size.

Remarks

Call the SetBLOBData procedure to set the parameter value from the memory buffer. After this procedure call the DataType property is assigned to ftBLOB.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.14 TIBCPParams Class

Used to control TIBCPParam objects.

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

Unit

[IBC](#)

Syntax

```
TIBCPParams = class(TDAPParams);
```

Remarks

Use TIBCPParams to manage a list of TIBCPParam objects for an object that uses field parameters. For example, TIBCStoredProc objects and TIBCQuery objects use TIBCPParams

objects to create and access their parameters.

Inheritance Hierarchy

[TDAParams](#)

TIBCPARAMS

See Also

- [TIBCPARAM](#)
- [TCustomDASQL.Params](#)
- [TCustomDADataset.Params](#)
- [TIBCSQL.Params](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.14.1 Members

[TIBCPARAMS](#) class overview.

Properties

Name	Description
Items	Used to iterate through all field parameters.

Methods

Name	Description
FindParam	Searches a parameter with the name passed in Value.
ParamByName	Searches a parameter with the name passed in Value.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.14.2 Properties

Properties of the **TIBCPParams** class.

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

Public

Name	Description
Items	Used to iterate through all field parameters.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.14.2.1 Items Property(Indexer)

Used to iterate through all field parameters.

Class

[TIBCPParams](#)

Syntax

```
property Items[Index: integer]: TIBCPParam; default;
```

Parameters

Index

Holds the index in the range 0..Count - 1.

Remarks

Use the Items property to iterate through all field parameters. Index identifies the index in the range 0..Count - 1. Items can reference a particular parameter by its index, but the ParamByName method is preferred, so as to avoid depending on the order of the parameters.

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.11.1.14.3 Methods

Methods of the **TIBCPParams** class.

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

Public

Name	Description
FindParam	Searches a parameter with the name passed in Value.
ParamByName	Searches a parameter with the name passed in Value.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.14.3.1 FindParam Method

Searches a parameter with the name passed in Value.

Class

[TIBCPParams](#)

Syntax

```
function FindParam(const value: string): TIBCPParam;
```

Parameters

Value

Holds the parameter name.

Return Value

the parameter, if a match was found.

Remarks

Call the FindParam method to find a parameter with the name passed in Value. If a match is found, FindParam returns the parameter. Otherwise, it returns nil. Use this method rather than a direct reference to the Items property to avoid depending on the order of the entries.

To locate more than one parameter at a time by name, use the GetParamList method instead. To get only the value of a named parameter, use the ParamValues property.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.14.3.2 ParamByName Method

Searches a parameter with the name passed in Value.

Class

[TIBCPARAMS](#)

Syntax

```
function ParamByName(const Value: string): TIBCPARAM;
```

Parameters

Value

Holds the parameter name.

Return Value

the parameter, if a match was found.

Remarks

Call the ParamByName method to find a parameter with the name passed in Value. If a match is found, ParamByName returns the parameter. Otherwise, an exception is raised. Use this method rather than a direct reference to the Items property to avoid depending on the order of the entries.

To locate a parameter by name without raising an exception if the parameter is not found, use the FindParam method.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.15 TIBCQuery 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 [TIBCQuery](#) members.

Unit

[IBC](#)

Syntax

```
TIBCQuery = class(TCustomIBCQuery);
```

Remarks

TIBCQuery is a direct descendant of the [TCustomIBCDataSet](#) component. It publishes most of its inherited properties and events so that they can be manipulated at design-time.

Use TIBCQuery to perform fetching, insertion, deletion and update of record by dynamically generated SQL statements. TIBCQuery 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, TIBCQuery will retrieve primary keys for UpdatingTable from metadata. TIBCQuery can automatically update only one table. Updating table is defined by the UpdatingTable property if this property is set. Otherwise, the table a field of which is the first field in the field list in the SELECT clause is used as an updating table.

The SQLInsert, SQLDelete, SQLUpdate, SQLRefresh properties support automatic binding of parameters which have identical names to fields captions. To retrieve the value of a field as it was before the operation use the field name with the 'OLD_' prefix. This is especially useful when doing field comparisons in the WHERE clause of the statement. Use the [TCustomDADataset.BeforeUpdateExecute](#) event to assign the value to additional parameters and the [TCustomDADataset.AfterUpdateExecute](#) event to read them.

Inheritance Hierarchy

[TMemDataSet](#)

[TCustomDADataset](#)

[TCustomIBCDataSet](#)

[TCustomIBCQuery](#)

TIBCQuery

See Also

- [Updating Data with IBDAC Dataset Components](#)
- [Master/Detail Relationships](#)
- [TIBCStoredProc](#)
- [TIBCTable](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.15.1 Members

[TIBCQuery](#) class overview.

Properties

Name	Description
AutoCommit (inherited from TCustomIBCDataSet)	Used to automatically commit each update, insert or delete statement by database server.
BaseSQL (inherited from TCustomDADataset)	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions (inherited from TCustomDADataset)	Used to add WHERE conditions to a query
Connection (inherited from TCustomIBCDataSet)	Used to specify the connection in which the dataset will be executed.

Cursor (inherited from TCustomIBCDataSet)	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
DataTypeMap (inherited from TCustomDADataset)	Used to set data type mapping rules
Debug (inherited from TCustomDADataset)	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields (inherited from TCustomDADataset)	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
Disconnected (inherited from TCustomDADataset)	Used to keep dataset opened after connection is closed.
DMLRefresh (inherited from TCustomIBCDataSet)	Used to refresh record by the RETURNING clause when insert is performed.
Encryption (inherited from TCustomIBCDataSet)	Used to specify encryption options in a dataset.
ExplainPlan (inherited from TCustomIBCDataSet)	Used to obtain the query execution plan for Table, Query, and SQL components.
FetchAll	Defines whether to request all records of the query from database server when the dataset is being opened.
FetchRows (inherited from TCustomDADataset)	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL (inherited from TCustomDADataset)	Used to change the WHERE clause of SELECT statement and reopen a query.
FinalSQL (inherited from TCustomDADataset)	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with

	expanded macros.
GeneratorMode (inherited from TCustomIBCDataSet)	Used to specify which method is used internally to generate a sequenced field.
GeneratorStep (inherited from TCustomIBCDataSet)	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
Handle (inherited from TCustomIBCDataSet)	Used to specify the handle for the SQL statement of TCustomIBCDataSet.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
IsQuery (inherited from TCustomIBCDataSet)	Used to check if the SQL statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
KeyFields (inherited from TCustomDADDataSet)	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
KeyGenerator (inherited from TCustomIBCDataSet)	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
LockMode	Used to specify what kind of lock will be performed when editing a record.
MacroCount (inherited from TCustomDADDataSet)	Used to get the number of macros associated with the

	Macros property.
Macros (inherited from TCustomDADataset)	Makes it possible to change SQL queries easily.
MasterFields (inherited from TCustomDADataset)	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)	Used to specify the data source component which binds current dataset to the master one.
Options (inherited from TCustomIBCDataSet)	Used to specify the behaviour of the TCustomIBCDataSet object.
ParamCheck (inherited from TCustomDADataset)	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
ParamCount (inherited from TCustomDADataset)	Used to indicate how many parameters are there in the Params property.
Params (inherited from TCustomDADataset)	Used to view and set parameter names, values, and data types dynamically.
Plan (inherited from TCustomIBCDataSet)	Used to get or set the PLAN clause of the SELECT statement.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
ReadOnly (inherited from TCustomDADataset)	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions (inherited from TCustomDADataset)	Used to indicate when the editing record is refreshed.
RowsAffected (inherited from TCustomDADataset)	Used to indicate the number of rows which were inserted, updated, or deleted during

	the last query operation.
RowsDeleted (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were deleted during the last query operation.
RowsFetched (inherited from TCustomIBCDataSet)	Used to get the number of the currently fetched rows.
RowsInserted (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were inserted during the last query operation.
RowsUpdated (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were updated during the last query operation.
SmartFetch (inherited from TCustomIBCDataSet)	The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.
SQL (inherited from TCustomDADDataSet)	Used to provide a SQL statement that a query component executes when its Open method is called.
SQLDelete (inherited from TCustomDADDataSet)	Used to specify a SQL statement that will be used when applying a deletion to a record.
SQLInsert (inherited from TCustomDADDataSet)	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
SQLLock (inherited from TCustomDADDataSet)	Used to specify a SQL statement that will be used to perform a record lock.
SQLRecCount (inherited from TCustomDADDataSet)	Used to specify the SQL statement that is used to get the record count when opening a dataset.
SQLRefresh (inherited from TCustomDADDataSet)	Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADDataSet.RefreshRecord procedure.

SQLType (inherited from TCustomIBCDataSet)	Used to get the typecode of the SQL statement being processed by the InterBase database server.
SQLUpdate (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying an update to a dataset.
Transaction (inherited from TCustomIBCDataSet)	Used to determine the transaction under which the query of this dataset executes.
UniDirectional (inherited from TCustomDADataset)	Used if an application does not need bidirectional access to records in the result set.
UpdateObject (inherited from TCustomIBCDataSet)	Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.
UpdateRecordTypes (inherited from TMemDataSet)	Used to indicate the update status for the current record when cached updates are enabled.
UpdatesPending (inherited from TMemDataSet)	Used to check the status of the cached updates buffer.
UpdateTransaction (inherited from TCustomIBCDataSet)	Used to get or set the transaction for modifying a dataset.
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.

Methods

Name	Description
AddWhere (inherited from TCustomDADataset)	Adds condition to the WHERE clause of SELECT statement in the SQL

	property.
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
BreakExec (inherited from TCustomDADataset)	Breaks execution of the SQL statement on the server.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
CreateBlobStream (inherited from TCustomDADataset)	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
CreateProcCall (inherited from TCustomIBCDataSet)	Assigns PL/SQL block that calls stored procedure to the SQL property.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
DeleteWhere (inherited from TCustomDADataset)	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
Execute (inherited from TCustomDADataset)	Overloaded. Executes a SQL statement on the server.
Executing (inherited from TCustomDADataset)	Indicates whether SQL statement is still being executed.

Fetched (inherited from TCustomDADataset)	Used to find out whether TCustomDADataset has fetched all rows.
Fetching (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is still fetching rows.
FetchingAll (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is fetching all rows to the end.
FindKey (inherited from TCustomDADataset)	Searches for a record which contains specified field values.
FindMacro (inherited from TCustomDADataset)	Finds a macro with the specified name.
FindNearest (inherited from TCustomDADataset)	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.
FindParam (inherited from TCustomIBCDataset)	Determines if a parameter with the specified name exists in a dataset.
GetArray (inherited from TCustomIBCDataset)	Retrieves a TIBCArrary object for a field when only its name is known.
GetBlob (inherited from TCustomIBCDataset)	Retrieves a TIBCBlob object for a field when only its name is known.
GetDataType (inherited from TCustomDADataset)	Returns internal field types defined in the MemData and accompanying modules.
GetFieldObject (inherited from TCustomDADataset)	Returns a multireference shared object from field.
GetFieldPrecision (inherited from TCustomDADataset)	Retrieves the precision of a number field.
GetFieldScale (inherited from TCustomDADataset)	Retrieves the scale of a number field.
GetKeyFieldNames (inherited from TCustomDADataset)	Provides a list of available key field names.
GetOrderBy (inherited from TCustomDADataset)	Retrieves an ORDER BY clause from a SQL statement.

GotoCurrent (inherited from TCustomDADataset)	Sets the current record in this dataset similar to the current record in another dataset.
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.
Lock (inherited from TCustomDADataset)	Locks the current record.
MacroByName (inherited from TCustomDADataset)	Finds a macro with the specified name.
ParamByName (inherited from TCustomIBCDataSet)	Called to set or use parameter information for a specific parameter based on its name.
Prepare (inherited from TCustomDADataset)	Allocates, opens, and parses cursor for a query.
RefreshRecord (inherited from TCustomDADataset)	Actualizes field values for the current record.
RestoreSQL (inherited from TCustomDADataset)	Restores the SQL property modified by AddWhere and SetOrderBy.
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.
SaveSQL (inherited from TCustomDADataset)	Saves the SQL property value to BaseSQL.
SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
SetOrderBy (inherited from TCustomDADataset)	Builds an ORDER BY clause of a SELECT statement.

SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
SQLSaved (inherited from TCustomDADataset)	Determines if the SQL property value was saved to the BaseSQL property.
Unlock (inherited from TCustomDADataset)	Releases a record lock.
UnPrepare (inherited from TMemDataSet)	Frees the resources allocated for a previously prepared query on the server and client sides.
UpdateResult (inherited from TMemDataSet)	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
UpdateStatus (inherited from TMemDataSet)	Indicates the current update status for the dataset when cached updates are enabled.

Events

Name	Description
AfterExecute (inherited from TCustomDADataset)	Occurs after a component has executed a query to database.
AfterFetch (inherited from TCustomDADataset)	Occurs after dataset finishes fetching data from server.
AfterUpdateExecute (inherited from TCustomDADataset)	Occurs after executing insert, delete, update, lock and refresh operations.

BeforeFetch (inherited from TCustomDADataset)	Occurs before dataset is going to fetch block of records from the server.
BeforeUpdateExecute (inherited from TCustomDADataset)	Occurs before executing insert, delete, update, lock, and refresh operations.
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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.15.2 Properties

Properties of the **TIBCQuery** class.

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

Public

Name	Description
AutoCommit (inherited from TCustomIBCDataset)	Used to automatically commit each update, insert or delete statement by database server.
BaseSQL (inherited from TCustomDADataset)	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions (inherited from TCustomDADataset)	Used to add WHERE conditions to a query
Connection (inherited from TCustomIBCDataset)	Used to specify the connection in which the dataset will be executed.
Cursor (inherited from TCustomIBCDataset)	Used for positioned UPDATE and DELETE

	statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
DataTypeMap (inherited from TCustomDADataset)	Used to set data type mapping rules
Debug (inherited from TCustomDADataset)	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields (inherited from TCustomDADataset)	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
Disconnected (inherited from TCustomDADataset)	Used to keep dataset opened after connection is closed.
DMLRefresh (inherited from TCustomIBCDataSet)	Used to refresh record by the RETURNING clause when insert is performed.
Encryption (inherited from TCustomIBCDataSet)	Used to specify encryption options in a dataset.
ExplainPlan (inherited from TCustomIBCDataSet)	Used to obtain the query execution plan for Table, Query, and SQL components.
FetchRows (inherited from TCustomDADataset)	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL (inherited from TCustomDADataset)	Used to change the WHERE clause of SELECT statement and reopen a query.
FinalSQL (inherited from TCustomDADataset)	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
GeneratorMode (inherited from TCustomIBCDataSet)	Used to specify which method is used internally to generate a sequenced field.
GeneratorStep (inherited from TCustomIBCDataSet)	Used to set the increment for

	increasing or decreasing current generator value when using the automatic key field value generation feature.
Handle (inherited from TCustomIBCDataSet)	Used to specify the handle for the SQL statement of TCustomIBCDataSet.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
IsQuery (inherited from TCustomIBCDataSet)	Used to check if the SQL statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
KeyFields (inherited from TCustomDADDataSet)	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
KeyGenerator (inherited from TCustomIBCDataSet)	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
MacroCount (inherited from TCustomDADDataSet)	Used to get the number of macros associated with the Macros property.
Macros (inherited from TCustomDADDataSet)	Makes it possible to change SQL queries easily.
MasterFields (inherited from TCustomDADDataSet)	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)	Used to specify the data source component which binds current dataset to the master one.
Options (inherited from TCustomIBCDataset)	Used to specify the behaviour of the TCustomIBCDataset object.
ParamCheck (inherited from TCustomDADataset)	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
ParamCount (inherited from TCustomDADataset)	Used to indicate how many parameters are there in the Params property.
Params (inherited from TCustomDADataset)	Used to view and set parameter names, values, and data types dynamically.
Plan (inherited from TCustomIBCDataset)	Used to get or set the PLAN clause of the SELECT statement.
Prepared (inherited from TMemDataset)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataset)	Indicates whether a range is applied to a dataset.
ReadOnly (inherited from TCustomDADataset)	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions (inherited from TCustomDADataset)	Used to indicate when the editing record is refreshed.
RowsAffected (inherited from TCustomDADataset)	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
RowsDeleted (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were deleted during the last query operation.
RowsFetched (inherited from TCustomIBCDataset)	Used to get the number of the currently fetched rows.
RowsInserted (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were inserted

	during the last query operation.
RowsUpdated (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were updated during the last query operation.
SmartFetch (inherited from TCustomIBCDataSet)	The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.
SQL (inherited from TCustomDADDataSet)	Used to provide a SQL statement that a query component executes when its Open method is called.
SQLDelete (inherited from TCustomDADDataSet)	Used to specify a SQL statement that will be used when applying a deletion to a record.
SQLInsert (inherited from TCustomDADDataSet)	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
SQLLock (inherited from TCustomDADDataSet)	Used to specify a SQL statement that will be used to perform a record lock.
SQLRecCount (inherited from TCustomDADDataSet)	Used to specify the SQL statement that is used to get the record count when opening a dataset.
SQLRefresh (inherited from TCustomDADDataSet)	Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADDataSet.RefreshRecord procedure.
SQLType (inherited from TCustomIBCDataSet)	Used to get the typecode of the SQL statement being processed by the InterBase database server.
SQLUpdate (inherited from TCustomDADDataSet)	Used to specify a SQL statement that will be used when applying an update to a dataset.

Transaction (inherited from TCustomIBCDataSet)	Used to determine the transaction under which the query of this dataset executes.
UniDirectional (inherited from TCustomDADataset)	Used if an application does not need bidirectional access to records in the result set.
UpdateObject (inherited from TCustomIBCDataSet)	Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.
UpdateRecordTypes (inherited from TMemDataSet)	Used to indicate the update status for the current record when cached updates are enabled.
UpdatesPending (inherited from TMemDataSet)	Used to check the status of the cached updates buffer.
UpdateTransaction (inherited from TCustomIBCDataSet)	Used to get or set the transaction for modifying a dataset.

Published

Name	Description
FetchAll	Defines whether to request all records of the query from database server when the dataset is being opened.
LockMode	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

- [TIBCQuery Class](#)

- [TIBCQuery Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.15.2.1 FetchAll Property

Defines whether to request all records of the query from database server when the dataset is being opened.

Class

[TIBCQuery](#)

Syntax

```
property FetchAll: boolean;
```

Remarks

When set to True, all records of the query are requested from database server when the dataset is being opened. When set to False, records are retrieved when a data-aware component or a program requests it. If a query can return a lot of records, set this property to False if initial response time is important.

When the FetchAll property is False, the first call to [TMemDataSet.Locate](#) and [TMemDataSet.LocateEx](#) methods may take a lot of time to retrieve additional records to the client side.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.15.2.2 LockMode Property

Used to specify what kind of lock will be performed when editing a record.

Class

[TIBCQuery](#)

Syntax

```
property LockMode: TLockMode default TmNone;
```

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

- [TIBCStoredProc.LockMode](#)
- [TIBCTable.LockMode](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.15.2.3 UpdatingTable Property

Used to specify which table in a query is assumed to be the target for subsequent data-modification queries as a result of user incentive to insert, update or delete records.

Class

[TIBCQuery](#)

Syntax

```
property updatingTable: string;
```

Remarks

Use the UpdatingTable property to specify which table in a query is assumed to be the target for the subsequent data-modification queries as a result of user incentive to insert, update or delete records.

This property is used on Insert, Update, Delete or RefreshRecord (see also [TCustomIBCDDataSet.Options](#)) if appropriate SQL (SQLInsert, SQLUpdate or SQLDelete) is not provided.

If UpdatingTable is not set then the first table used in a query is assumed to be the target.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16 TIBCSQL Class

A component for executing SQL statements and calling stored procedures on the database server.

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

Unit

[IBC](#)

Syntax

```
TIBCSQL = class(TCustomDASQL);
```

Remarks

The TIBCSQL component is a direct descendant of the [TCustomDASQL](#) class.

Use The TIBCSQL component when a client application must execute SQL statement or the PL/SQL block, and call stored procedure on the database server. The SQL statement should not retrieve rows from the database.

Inheritance Hierarchy

[TCustomDASQL](#)

TIBCSQL

See Also

- [TIBCQuery](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.1 Members

[TIBCSQL](#) class overview.

Properties

Name	Description
ChangeCursor (inherited from TCustomDASQL)	Enables or disables changing screen cursor when executing commands in the NonBlocking mode.
Connection	Used to set or return the connection associated with the query.
Debug (inherited from TCustomDASQL)	Used to display the statement that is being executed and the values and types of its parameters.
DescribeParams	Used to specify whether to query TIBCPParam properties from the server when executing the TCustomDASQL.Prepare method.
ExplainPlan	Used to obtain the query execution plan for Table, Query, and SQL components.
FinalSQL (inherited from TCustomDASQL)	Used to return a SQL statement with expanded macros.
Handle	Used to specify the handle for the SQL statement of TIBCSQL.
MacroCount (inherited from TCustomDASQL)	Used to get the number of macros associated with the Macros property.
Macros (inherited from TCustomDASQL)	Makes it possible to change SQL queries easily.
ParamCheck (inherited from TCustomDASQL)	Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.
ParamCount (inherited from TCustomDASQL)	Indicates the number of parameters in the Params property.
Params	Contains the parameters for SQL statement.
ParamValues (inherited from TCustomDASQL)	Used to get or set the values

	of individual field parameters that are identified by name.
Plan	Used to get or set the PLAN clause of the SELECT statement.
Prepared (inherited from TCustomDASQL)	Used to indicate whether a query is prepared for execution.
RowsAffected (inherited from TCustomDASQL)	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
SQL (inherited from TCustomDASQL)	Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.
SQLType	Used to provide the typecode of the SQL statement being processed by the InterBase server.
Transaction	Used to set or return the transaction to be used by the component.

Methods

Name	Description
BreakExec (inherited from TCustomDASQL)	Breaks execution of an SQL statement on the server.
CreateProcCall	Assigns SQL block that calls stored procedure specified by Name to SQL property.
Execute (inherited from TCustomDASQL)	Overloaded. Executes a SQL statement on the server.
ExecuteNext	Provides data that is returned by SQL statement through the out parameters.
Executing (inherited from TCustomDASQL)	Checks whether TCustomDASQL still executes a SQL statement.

FindMacro (inherited from TCustomDASQL)	Finds a macro with the specified name.
FindParam	Searches a parameter with the specified name.
MacroByName (inherited from TCustomDASQL)	Finds a macro with the specified name.
ParamByName	Searches a parameter with the specified name.
Prepare (inherited from TCustomDASQL)	Allocates, opens, and parses cursor for a query.
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.

Events

Name	Description
AfterExecute (inherited from TCustomDASQL)	Occurs after a SQL statement has been executed.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.2 Properties

Properties of the **TIBCSQL** class.

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

Public

Name	Description
ChangeCursor (inherited from TCustomDASQL)	Enables or disables changing screen cursor when executing commands in the NonBlocking mode.
Debug (inherited from TCustomDASQL)	Used to display the statement that is being executed and the values and

	types of its parameters.
ExplainPlan	Used to obtain the query execution plan for Table, Query, and SQL components.
FinalSQL (inherited from TCustomDASQL)	Used to return a SQL statement with expanded macros.
Handle	Used to specify the handle for the SQL statement of TIBCSQL.
MacroCount (inherited from TCustomDASQL)	Used to get the number of macros associated with the Macros property.
Macros (inherited from TCustomDASQL)	Makes it possible to change SQL queries easily.
ParamCheck (inherited from TCustomDASQL)	Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.
ParamCount (inherited from TCustomDASQL)	Indicates the number of parameters in the Params property.
ParamValues (inherited from TCustomDASQL)	Used to get or set the values of individual field parameters that are identified by name.
Plan	Used to get or set the PLAN clause of the SELECT statement.
Prepared (inherited from TCustomDASQL)	Used to indicate whether a query is prepared for execution.
RowsAffected (inherited from TCustomDASQL)	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
SQL (inherited from TCustomDASQL)	Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.

SQLType	Used to provide the typecode of the SQL statement being processed by the InterBase server.
-------------------------	--

Published

Name	Description
Connection	Used to set or return the connection associated with the query.
DescribeParams	Used to specify whether to query TIBCPParam properties from the server when executing the TCustomDASQL.Prepare method.
Params	Contains the parameters for SQL statement.
Transaction	Used to set or return the transaction to be used by the component.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.2.1 Connection Property

Used to set or return the connection associated with the query.

Class

[TIBCSQL](#)

Syntax

```
property Connection: TIBConnection;
```

Remarks

Use the Connection property to set or return the connection associated with the query.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.2.2 DescribeParams Property

Used to specify whether to query [TIBCPParam](#) properties from the server when executing the [TCustomDASQL.Prepare](#) method.

Class

[TIBCSQL](#)

Syntax

```
property DescribeParams: boolean default False;
```

Remarks

Specifies whether to query [TIBCPParam](#) properties (Name, ParamType, DataType, Size, TableName) from the server when executing the [TCustomDASQL.Prepare](#) method. The default value is False.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.2.3 ExplainPlan Property

Used to obtain the query execution plan for Table, Query, and SQL components.

Class

[TIBCSQL](#)

Syntax

```
property ExplainPlan: string;
```

Remarks

Use the ExplainPlan property to obtain the query execution plan for Table, Query, and SQL

components.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.2.4 Handle Property

Used to specify the handle for the SQL statement of TIBCSQL.

Class

[TIBCSQL](#)

Syntax

```
property Handle: TISC_STMT_HANDLE;
```

Remarks

Use the Handle property to specify the handle for the SQL statement of TIBCSQL.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.2.5 Params Property

Contains the parameters for SQL statement.

Class

[TIBCSQL](#)

Syntax

```
property Params: TIBCParams stored False;
```

Remarks

The Params property is used to hold the parameters for SQL statement.

Access Params at runtime to view and set parameter names, values, and data types dynamically (at design time use the Parameters editor to set parameter information). Params is a zero-based array of parameter records. Index specifies the array element to access.

An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName.

See Also

- [TIBCPParam](#)
- [FindParam](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.2.6 Plan Property

Used to get or set the PLAN clause of the SELECT statement.

Class

[TIBCSQL](#)

Syntax

```
property Plan: string;
```

Remarks

Use the Plan property to get or set the PLAN clause of the SELECT statement.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.2.7 SQLType Property

Used to provide the typecode of the SQL statement being processed by the InterBase server.

Class

[TIBCSQL](#)

Syntax

```
property SQLType: integer;
```

Remarks

Use the `SQLType` property to get the typecode of the SQL statement being processed by the InterBase server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.2.8 Transaction Property

Used to set or return the transaction to be used by the component.

Class

[TIBCSQL](#)

Syntax

```
property Transaction: TIBCTransaction stored IsTransactionStored;
```

Remarks

Use the Transaction property to set or return the transaction to be used by the component.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.3 Methods

Methods of the **TIBCSQL** class.

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

Public

Name	Description
BreakExec (inherited from TCustomDASQL)	Breaks execution of an SQL statement on the server.
CreateProcCall	Assigns SQL block that calls stored procedure specified by Name to SQL property.
Execute (inherited from TCustomDASQL)	Overloaded. Executes a SQL statement on the server.
ExecuteNext	Provides data that is returned by SQL statement

	through the out parameters.
Executing (inherited from TCustomDASQL)	Checks whether TCustomDASQL still executes a SQL statement.
FindMacro (inherited from TCustomDASQL)	Finds a macro with the specified name.
FindParam	Searches a parameter with the specified name.
MacroByName (inherited from TCustomDASQL)	Finds a macro with the specified name.
ParamByName	Searches a parameter with the specified name.
Prepare (inherited from TCustomDASQL)	Allocates, opens, and parses cursor for a query.
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.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.3.1 CreateProcCall Method

Assigns SQL block that calls stored procedure specified by Name to SQL property.

Class

[TIBCSQL](#)

Syntax

```
procedure CreateProcCall(const Name: string);
```

Parameters

Name

Holds the stored procedure name.

Remarks

Call CreateProcCall to assign SQL block that calls stored procedure specified by Name to SQL property. Retrieves the information about parameters of the procedure from InterBase. After calling CreateProcCall you can execute stored procedure by Execute method.

See Also

- [TCustomDASQL.Execute](#)
- [TCustomDACConnection.ExecProc](#)
- [TIBCStoredProc](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.3.2 ExecuteNext Method

Provides data that is returned by SQL statement through the out parameters.

Class

[TIBCSQL](#)

Syntax

```
function ExecuteNext: boolean;
```

Return Value

True, when the data were read and False when it reaches the end of the dataset.

Remarks

Call the ExecuteNext method to get data that is returned by SQL statement through the out parameters. That can be select statement or execution of stored procedure that returns dataset. ExecuteNext returns True when the data were read and False when it reaches the end of dataset.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.3.3 FindParam Method

Searches a parameter with the specified name.

Class

[TIBCSQL](#)

Syntax

```
function FindParam(const value: string): TIBCPParam;
```

Parameters

Value

Holds the parameter name.

Return Value

the parameter, if a match was found.

Remarks

Call the FindParam method to find a parameter with the name passed in the Name argument. If a match is found, FindParam returns the parameter. Otherwise, it returns nil.

See Also

- [TIBCPParam](#)
- [ParamByName](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.16.3.4 ParamByName Method

Searches a parameter with the specified name.

Class

[TIBCSQL](#)

Syntax

```
function ParamByName(const value: string): TIBCPParam;
```

Parameters

Value

Holds the parameter name.

Return Value

the parameter, if a match was found.

Remarks

Call the ParamByName method to find a parameter with the name passed in Name argument.

If a match is found, ParamByName returns the parameter. Otherwise, an exception is raised.

See Also

- [TIBCPParam](#)
- [FindParam](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.17 TIBCSSLConnectionOptions Class

A class for setting up the SSL options.

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

Unit

[IBC](#)

Syntax

```
TIBCSSLConnectionOptions = class(TPersistent);
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.17.1 Members

[TIBCSSLConnectionOptions](#) class overview.

Properties

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

ClientCertFile	Holds the path to the client certificate file.
ClientPassPhrase	Holds the private key passphrase.
ClientPassPhraseFile	Holds the path to the text file containing the client private key passphrase.
Enabled	Enables or disables SSL connections.
ServerPublicFile	Holds the path to the certificate authority file.
ServerPublicPath	Holds the path to the directory with the certificate authority files.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.17.2 Properties

Properties of the **TIBCSSLConnectionOptions** class.

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

Published

Name	Description
ClientCertFile	Holds the path to the client certificate file.
ClientPassPhrase	Holds the private key passphrase.
ClientPassPhraseFile	Holds the path to the text file containing the client private key passphrase.
Enabled	Enables or disables SSL connections.
ServerPublicFile	Holds the path to the certificate authority file.
ServerPublicPath	Holds the path to the directory with the certificate authority files.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.17.2.1 ClientCertFile Property

Holds the path to the client certificate file.

Class

[TIBCSSLConnectionOptions](#)

Syntax

```
property ClientCertFile: string;
```

Remarks

Use the ClientCertFile property to specify the path the client certificate file. The file must be in the PEM format and contain both the client certificate and the private key.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.17.2.2 ClientPassPhrase Property

Holds the private key passphrase.

Class

[TIBCSSLConnectionOptions](#)

Syntax

```
property ClientPassPhrase: string;
```

Remarks

Use the ClientPassPhrase property to specify the private key passphrase. You can use either

this property or the [ClientPassPhraseFile](#) property.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.17.2.3 ClientPassPhraseFile Property

Holds the path to the text file containing the client private key passphrase.

Class

[TIBCSSLConnectionOptions](#)

Syntax

```
property ClientPassPhraseFile: string;
```

Remarks

Use the ClientPassPhraseFile property to specify the path to the text file containing the client private key passphrase. You can use either this property or the [ClientPassPhrase](#) property.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.17.2.4 Enabled Property

Enables or disables SSL connections.

Class

[TIBCSSLConnectionOptions](#)

Syntax

```
property Enabled: boolean;
```

Remarks

Use the Enabled property to enable or disable SSL connections.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.17.2.5 ServerPublicFile Property

Holds the path to the certificate authority file.

Class

[TIBCSSLConnectionOptions](#)

Syntax

```
property ServerPublicFile: string;
```

Remarks

Use the ServerPublicFile property to specify the path to the CA certificate file in the PEM format.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.17.2.6 ServerPublicPath Property

Holds the path to the directory with the certificate authority files.

Class

[TIBCSSLConnectionOptions](#)

Syntax

```
property ServerPublicPath: string;
```

Remarks

Use the ServerPublicPath property to specify the path to the directory with the CA certificate files in the PEM format. Each file in the directory must contain only a single CA certificate and the files must be named by the hash of the subject name and extension of ".0". It is recommended that you use [ServerPublicFile](#) instead. If you specify both, ServerPublicFile will be used.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.18 TIBCStoredProc Class

A component for accessing and executing stored procedures and functions.

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

Unit

[IBC](#)

Syntax

```
TIBCStoredProc = class(TCustomIBCQuery);
```

Remarks

Use TIBCStoredProc to access stored procedures on the database server.

You need only to define the StoredProcName property, and the SQL statement to call the stored procedure will be generated automatically.

Use the Execute method at runtime to generate request that instructs server to execute procedure and PrepareSQL to describe parameters at run time

Inheritance Hierarchy

[TMemDataSet](#)

[TCustomDADataset](#)

[TCustomIBCDataSet](#)

[TCustomIBCQuery](#)

TIBCStoredProc

See Also

- [TIBCQuery](#)
- [TIBCSQL](#)
- [Updating Data with IBDAC Dataset Components](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.18.1 Members

[TIBCStoredProc](#) class overview.

Properties

Name	Description
AutoCommit (inherited from TCustomIBCDataSet)	Used to automatically commit each update, insert or delete statement by database server.
BaseSQL (inherited from TCustomDADataset)	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions (inherited from TCustomDADataset)	Used to add WHERE conditions to a query
Connection (inherited from TCustomIBCDataSet)	Used to specify the connection in which the dataset will be executed.
Cursor (inherited from TCustomIBCDataSet)	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
DataTypeMap (inherited from TCustomDADataset)	Used to set data type mapping rules
Debug (inherited from TCustomDADataset)	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields (inherited from TCustomDADataset)	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
Disconnected (inherited from TCustomDADataset)	Used to keep dataset opened after connection is closed.
DMLRefresh (inherited from TCustomIBCDataSet)	Used to refresh record by the RETURNING clause

	when insert is performed.
Encryption (inherited from TCustomIBCDataSet)	Used to specify encryption options in a dataset.
ExplainPlan (inherited from TCustomIBCDataSet)	Used to obtain the query execution plan for Table, Query, and SQL components.
FetchAll (inherited from TCustomIBCDataSet)	Used to retrieve all records in a dataset.
FetchRows (inherited from TCustomDADataset)	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL (inherited from TCustomDADataset)	Used to change the WHERE clause of SELECT statement and reopen a query.
FinalSQL (inherited from TCustomDADataset)	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
GeneratorMode (inherited from TCustomIBCDataSet)	Used to specify which method is used internally to generate a sequenced field.
GeneratorStep (inherited from TCustomIBCDataSet)	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
Handle (inherited from TCustomIBCDataSet)	Used to specify the handle for the SQL statement of TCustomIBCDataSet.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
IsQuery (inherited from TCustomIBCDataSet)	Used to check if the SQL statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
KeyFields (inherited from TCustomDADataset)	Used to build SQL statements for the SQLDelete, SQLInsert, and

	SQLUpdate properties if they were empty before updating the database.
KeyGenerator (inherited from TCustomIBCDataSet)	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
LockMode	Used to specify what kind of lock will be performed when editing a record.
MacroCount (inherited from TCustomDADDataSet)	Used to get the number of macros associated with the Macros property.
Macros (inherited from TCustomDADDataSet)	Makes it possible to change SQL queries easily.
MasterFields (inherited from TCustomDADDataSet)	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 TCustomDADDataSet)	Used to specify the data source component which binds current dataset to the master one.
Options (inherited from TCustomIBCDataSet)	Used to specify the behaviour of the TCustomIBCDataSet object.
ParamCheck (inherited from TCustomDADDataSet)	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
ParamCount (inherited from TCustomDADDataSet)	Used to indicate how many parameters are there in the

	Params property.
Params (inherited from TCustomDADataset)	Used to view and set parameter names, values, and data types dynamically.
Plan (inherited from TCustomIBCDataset)	Used to get or set the PLAN clause of the SELECT statement.
Prepared (inherited from TMemDataset)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataset)	Indicates whether a range is applied to a dataset.
ReadOnly (inherited from TCustomDADataset)	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions (inherited from TCustomDADataset)	Used to indicate when the editing record is refreshed.
RowsAffected (inherited from TCustomDADataset)	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
RowsDeleted (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were deleted during the last query operation.
RowsFetched (inherited from TCustomIBCDataset)	Used to get the number of the currently fetched rows.
RowsInserted (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were inserted during the last query operation.
RowsUpdated (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were updated during the last query operation.
SmartFetch (inherited from TCustomIBCDataset)	The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.
SQL (inherited from TCustomDADataset)	Used to provide a SQL statement that a query component executes when its Open method is called.

SQLDelete (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying a deletion to a record.
SQLInsert (inherited from TCustomDADataset)	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
SQLLock (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to perform a record lock.
SQLRecCount (inherited from TCustomDADataset)	Used to specify the SQL statement that is used to get the record count when opening a dataset.
SQLRefresh (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.
SQLType (inherited from TCustomIBCDataSet)	Used to get the typecode of the SQL statement being processed by the InterBase database server.
SQLUpdate (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying an update to a dataset.
StoredProcName	Used to specify the name of the stored procedure to call on the server.
Transaction (inherited from TCustomIBCDataSet)	Used to determine the transaction under which the query of this dataset executes.
UniDirectional (inherited from TCustomDADataset)	Used if an application does not need bidirectional access to records in the result set.
UpdateObject (inherited from TCustomIBCDataSet)	Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.

UpdateRecordTypes (inherited from TMemDataSet)	Used to indicate the update status for the current record when cached updates are enabled.
UpdatesPending (inherited from TMemDataSet)	Used to check the status of the cached updates buffer.
UpdateTransaction (inherited from TCustomIBCDataset)	Used to get or set the transaction for modifying a dataset.

Methods

Name	Description
AddWhere (inherited from TCustomDADataset)	Adds condition to the WHERE clause of SELECT statement in the SQL property.
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
BreakExec (inherited from TCustomDADataset)	Breaks execution of the SQL statement on the server.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
CreateBlobStream (inherited from TCustomDADataset)	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
CreateProcCall (inherited from TCustomIBCDataset)	Assigns PL/SQL block that calls stored procedure to the SQL property.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.

DeleteWhere (inherited from TCustomDADataset)	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
ExecProc	Executes a SQL statement on the server.
Execute (inherited from TCustomDADataset)	Overloaded. Executes a SQL statement on the server.
Executing (inherited from TCustomDADataset)	Indicates whether SQL statement is still being executed.
Fetched (inherited from TCustomDADataset)	Used to find out whether TCustomDADataset has fetched all rows.
Fetching (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is still fetching rows.
FetchingAll (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is fetching all rows to the end.
FindKey (inherited from TCustomDADataset)	Searches for a record which contains specified field values.
FindMacro (inherited from TCustomDADataset)	Finds a macro with the specified name.
FindNearest (inherited from TCustomDADataset)	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.
FindParam (inherited from TCustomIBCDataset)	Determines if a parameter with the specified name exists in a dataset.
GetArray (inherited from TCustomIBCDataset)	Retrieves a TIBCArry object for a field when only its name is known.

GetBlob (inherited from TCustomIBCDataset)	Retrieves a TIBCBlob object for a field when only its name is known.
GetDataType (inherited from TCustomDADataset)	Returns internal field types defined in the MemData and accompanying modules.
GetFieldObject (inherited from TCustomDADataset)	Returns a multireference shared object from field.
GetFieldPrecision (inherited from TCustomDADataset)	Retrieves the precision of a number field.
GetFieldScale (inherited from TCustomDADataset)	Retrieves the scale of a number field.
GetKeyFieldNames (inherited from TCustomDADataset)	Provides a list of available key field names.
GetOrderBy (inherited from TCustomDADataset)	Retrieves an ORDER BY clause from a SQL statement.
GotoCurrent (inherited from TCustomDADataset)	Sets the current record in this dataset similar to the current record in another dataset.
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.
Lock (inherited from TCustomDADataset)	Locks the current record.
MacroByName (inherited from TCustomDADataset)	Finds a macro with the specified name.
ParamByName (inherited from TCustomIBCDataset)	Called to set or use parameter information for a specific parameter based on its name.
Prepare	Describes the stored procedure parameters.
PrepareSQL	Describes the stored procedure parameters.

RefreshRecord (inherited from TCustomDADataset)	Actualizes field values for the current record.
RestoreSQL (inherited from TCustomDADataset)	Restores the SQL property modified by AddWhere and SetOrderBy.
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.
SaveSQL (inherited from TCustomDADataset)	Saves the SQL property value to BaseSQL.
SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
SetOrderBy (inherited from TCustomDADataset)	Builds an ORDER BY clause of a SELECT statement.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
SQLSaved (inherited from TCustomDADataset)	Determines if the SQL property value was saved to the BaseSQL property.
UnLock (inherited from TCustomDADataset)	Releases a record lock.
UnPrepare (inherited from TMemDataSet)	Frees the resources allocated for a previously prepared query on the server and client sides.
UpdateResult (inherited from TMemDataSet)	Reads the status of the latest call to the

	ApplyUpdates method while cached updates are enabled.
UpdateStatus (inherited from TMemDataSet)	Indicates the current update status for the dataset when cached updates are enabled.

Events

Name	Description
AfterExecute (inherited from TCustomDADataset)	Occurs after a component has executed a query to database.
AfterFetch (inherited from TCustomDADataset)	Occurs after dataset finishes fetching data from server.
AfterUpdateExecute (inherited from TCustomDADataset)	Occurs after executing insert, delete, update, lock and refresh operations.
BeforeFetch (inherited from TCustomDADataset)	Occurs before dataset is going to fetch block of records from the server.
BeforeUpdateExecute (inherited from TCustomDADataset)	Occurs before executing insert, delete, update, lock, and refresh operations.
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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.18.2 Properties

Properties of the **TIBCStoredProc** class.

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

Public

Name	Description
AutoCommit (inherited from TCustomIBCDataSet)	Used to automatically commit each update, insert or delete statement by database server.
BaseSQL (inherited from TCustomDADDataSet)	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions (inherited from TCustomDADDataSet)	Used to add WHERE conditions to a query
Connection (inherited from TCustomIBCDataSet)	Used to specify the connection in which the dataset will be executed.
Cursor (inherited from TCustomIBCDataSet)	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
DataTypeMap (inherited from TCustomDADDataSet)	Used to set data type mapping rules
Debug (inherited from TCustomDADDataSet)	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields (inherited from TCustomDADDataSet)	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
Disconnected (inherited from TCustomDADDataSet)	Used to keep dataset opened after connection is closed.
DMLRefresh (inherited from TCustomIBCDataSet)	Used to refresh record by the RETURNING clause when insert is performed.
Encryption (inherited from TCustomIBCDataSet)	Used to specify encryption options in a dataset.

ExplainPlan (inherited from TCustomIBCDataSet)	Used to obtain the query execution plan for Table, Query, and SQL components.
FetchAll (inherited from TCustomIBCDataSet)	Used to retrieve all records in a dataset.
FetchRows (inherited from TCustomDADataset)	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL (inherited from TCustomDADataset)	Used to change the WHERE clause of SELECT statement and reopen a query.
FinalSQL (inherited from TCustomDADataset)	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
GeneratorMode (inherited from TCustomIBCDataSet)	Used to specify which method is used internally to generate a sequenced field.
GeneratorStep (inherited from TCustomIBCDataSet)	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
Handle (inherited from TCustomIBCDataSet)	Used to specify the handle for the SQL statement of TCustomIBCDataSet.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
IsQuery (inherited from TCustomIBCDataSet)	Used to check if the SQL statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
KeyFields (inherited from TCustomDADataset)	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.

KeyGenerator (inherited from TCustomIBCDataSet)	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
MacroCount (inherited from TCustomDADataset)	Used to get the number of macros associated with the Macros property.
Macros (inherited from TCustomDADataset)	Makes it possible to change SQL queries easily.
MasterFields (inherited from TCustomDADataset)	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)	Used to specify the data source component which binds current dataset to the master one.
Options (inherited from TCustomIBCDataSet)	Used to specify the behaviour of the TCustomIBCDataSet object.
ParamCheck (inherited from TCustomDADataset)	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
ParamCount (inherited from TCustomDADataset)	Used to indicate how many parameters are there in the Params property.
Params (inherited from TCustomDADataset)	Used to view and set parameter names, values, and data types dynamically.
Plan (inherited from TCustomIBCDataSet)	Used to get or set the PLAN clause of the SELECT

	statement.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
ReadOnly (inherited from TCustomDADataset)	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions (inherited from TCustomDADataset)	Used to indicate when the editing record is refreshed.
RowsAffected (inherited from TCustomDADataset)	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
RowsDeleted (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were deleted during the last query operation.
RowsFetched (inherited from TCustomIBCDataSet)	Used to get the number of the currently fetched rows.
RowsInserted (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were inserted during the last query operation.
RowsUpdated (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were updated during the last query operation.
SmartFetch (inherited from TCustomIBCDataSet)	The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.
SQL (inherited from TCustomDADataset)	Used to provide a SQL statement that a query component executes when its Open method is called.
SQLDelete (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying a deletion to a record.
SQLInsert (inherited from TCustomDADataset)	Used to specify the SQL statement that will be used

	when applying an insertion to a dataset.
SQLLock (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to perform a record lock.
SQLRecCount (inherited from TCustomDADataset)	Used to specify the SQL statement that is used to get the record count when opening a dataset.
SQLRefresh (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.
SQLType (inherited from TCustomIBCDataSet)	Used to get the typecode of the SQL statement being processed by the InterBase database server.
SQLUpdate (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying an update to a dataset.
Transaction (inherited from TCustomIBCDataSet)	Used to determine the transaction under which the query of this dataset executes.
UniDirectional (inherited from TCustomDADataset)	Used if an application does not need bidirectional access to records in the result set.
UpdateObject (inherited from TCustomIBCDataSet)	Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.
UpdateRecordTypes (inherited from TMemDataSet)	Used to indicate the update status for the current record when cached updates are enabled.
UpdatesPending (inherited from TMemDataSet)	Used to check the status of the cached updates buffer.
UpdateTransaction (inherited from	Used to get or set the transaction for modifying a dataset.

[TCustomIBCDataset](#)

Published

Name	Description
<u>LockMode</u>	Used to specify what kind of lock will be performed when editing a record.
<u>StoredProcName</u>	Used to specify the name of the stored procedure to call on the server.

See Also

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

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.11.1.18.2.1 LockMode Property

Used to specify what kind of lock will be performed when editing a record.

Class

[TIBCStoredProc](#)

Syntax

```
property LockMode: TLockMode default ImNone;
```

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

- [TIBCQuery.LockMode](#)
- [TIBCTable.LockMode](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.18.2.2 StoredProcName Property

Used to specify the name of the stored procedure to call on the server.

Class

[TIBCStoredProc](#)

Syntax

```
property StoredProcName: string;
```

Remarks

Use the StoredProcName property to specify the name of the stored procedure to call on the server. If StoredProcName does not match the name of an existing stored procedure on the server, then when the application attempts to prepare the procedure prior to execution, an exception is raised.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.18.3 Methods

Methods of the **TIBCStoredProc** class.

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

Public

Name	Description
AddWhere (inherited from TCustomDADataset)	Adds condition to the WHERE clause of SELECT

	statement in the SQL property.
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
BreakExec (inherited from TCustomDADataset)	Breaks execution of the SQL statement on the server.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
CreateBlobStream (inherited from TCustomDADataset)	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
CreateProcCall (inherited from TCustomIBCDataSet)	Assigns PL/SQL block that calls stored procedure to the SQL property.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
DeleteWhere (inherited from TCustomDADataset)	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
ExecProc	Executes a SQL statement on the server.
Execute (inherited from TCustomDADataset)	Overloaded. Executes a SQL statement on the server.

Executing (inherited from TCustomDADataset)	Indicates whether SQL statement is still being executed.
Fetched (inherited from TCustomDADataset)	Used to find out whether TCustomDADataset has fetched all rows.
Fetching (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is still fetching rows.
FetchingAll (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is fetching all rows to the end.
FindKey (inherited from TCustomDADataset)	Searches for a record which contains specified field values.
FindMacro (inherited from TCustomDADataset)	Finds a macro with the specified name.
FindNearest (inherited from TCustomDADataset)	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.
FindParam (inherited from TCustomIBCDataSet)	Determines if a parameter with the specified name exists in a dataset.
GetArray (inherited from TCustomIBCDataSet)	Retrieves a TIBCArray object for a field when only its name is known.
GetBlob (inherited from TCustomIBCDataSet)	Retrieves a TIBCBlob object for a field when only its name is known.
GetDataType (inherited from TCustomDADataset)	Returns internal field types defined in the MemData and accompanying modules.
GetFieldObject (inherited from TCustomDADataset)	Returns a multireference shared object from field.
GetFieldPrecision (inherited from TCustomDADataset)	Retrieves the precision of a number field.
GetFieldScale (inherited from TCustomDADataset)	Retrieves the scale of a number field.
GetKeyFieldNames (inherited from TCustomDADataset)	Provides a list of available key field names.

GetOrderBy (inherited from TCustomDADataset)	Retrieves an ORDER BY clause from a SQL statement.
GotoCurrent (inherited from TCustomDADataset)	Sets the current record in this dataset similar to the current record in another dataset.
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.
Lock (inherited from TCustomDADataset)	Locks the current record.
MacroByName (inherited from TCustomDADataset)	Finds a macro with the specified name.
ParamByName (inherited from TCustomIBCDataSet)	Called to set or use parameter information for a specific parameter based on its name.
Prepare	Describes the stored procedure parameters.
PrepareSQL	Describes the stored procedure parameters.
RefreshRecord (inherited from TCustomDADataset)	Actualizes field values for the current record.
RestoreSQL (inherited from TCustomDADataset)	Restores the SQL property modified by AddWhere and SetOrderBy.
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.
SaveSQL (inherited from TCustomDADataset)	Saves the SQL property value to BaseSQL.
SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML

	format compatible with ADO format.
SetOrderBy (inherited from TCustomDADataset)	Builds an ORDER BY clause of a SELECT statement.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
SQLSaved (inherited from TCustomDADataset)	Determines if the SQL property value was saved to the BaseSQL property.
UnLock (inherited from TCustomDADataset)	Releases a record lock.
UnPrepare (inherited from TMemDataSet)	Frees the resources allocated for a previously prepared query on the server and client sides.
UpdateResult (inherited from TMemDataSet)	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
UpdateStatus (inherited from TMemDataSet)	Indicates the current update status for the dataset when cached updates are enabled.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.18.3.1 ExecProc Method

Executes a SQL statement on the server.

Class

[TIBCStoredProc](#)

Syntax

```
procedure ExecProc;
```

Remarks

ExecProc is similar to the [TCustomDADataset.Execute](#) method. It is included for compatibility with TStoredProc.

See Also

- [TCustomDADataset.Execute](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.18.3.2 Prepare Method

Describes the stored procedure parameters.

Class

[TIBCStoredProc](#)

Syntax

```
procedure Prepare; override;
```

Remarks

Call the Prepare method to describe the parameters of stored procedure. You can define parameters at design time if ParametersEditor is opened. Prepare method prepares the EXECUTE PROCEDURE statement. To prepare the SELECT statement use the [PrepareSQL](#) method.

See Also

- [PrepareSQL](#)
- [TCustomDADataset.Execute](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.18.3.3 PrepareSQL Method

Describes the stored procedure parameters.

Class

[TIBCToredProc](#)

Syntax

```
procedure PrepareSQL(IsQuery: boolean = False);
```

Parameters

IsQuery

True, if the SELECT statemnt should be prepared.

Remarks

Use the PrepareSQL method to describe the parameters of stored procedure. The Execute or Open method calls it automatically if it is necessary. You can define the parameters at design time if ParametersEditor is opened. Set IsQuery parameter to True to prepare SELECT statement. Set it to False or omit it to prepare EXECUTE PROCEDURE statement.

See Also

- [Prepare](#)
- [TCustomDADataset.Execute](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.19 TIBCTable Class

A component for retrieving and updating data in a single table without writing SQL statements.

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

Unit

[IBC](#)

Syntax

```
TIBCTable = class(TCustomIBCTable);
```

Remarks

The TIBCTable component allows retrieving and updating data in a single table without writing SQL statements. Use TIBCTable to access data in a table . Use the TableName property to specify table name. TIBCTable uses the KeyFields property to build SQL statements for updating table data. KeyFields is a string containing a semicolon-delimited list of the field names.

Inheritance Hierarchy

[TMemDataSet](#)

[TCustomDADataset](#)

[TCustomIBCDataset](#)

[TCustomIBCQuery](#)

[TCustomIBCTable](#)

TIBCTable

See Also

- [Updating Data with IBDAC Dataset Components](#)
- [Master/Detail Relationships](#)
- [TCustomIBCTable](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.19.1 Members

[TIBCTable](#) class overview.

Properties

Name	Description
AutoCommit (inherited from TCustomIBCDataSet)	Used to automatically commit each update, insert or delete statement by database server.
BaseSQL (inherited from TCustomDADataset)	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions (inherited from TCustomDADataset)	Used to add WHERE conditions to a query
Connection (inherited from TCustomIBCDataSet)	Used to specify the connection in which the dataset will be executed.
Cursor (inherited from TCustomIBCDataSet)	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
DataTypeMap (inherited from TCustomDADataset)	Used to set data type mapping rules
Debug (inherited from TCustomDADataset)	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields (inherited from TCustomDADataset)	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
Disconnected (inherited from TCustomDADataset)	Used to keep dataset opened after connection is closed.
DMLRefresh (inherited from TCustomIBCDataSet)	Used to refresh record by the RETURNING clause when insert is performed.
Encryption (inherited from TCustomIBCDataSet)	Used to specify encryption options in a dataset.
Exists (inherited from TCustomIBCTable)	Indicates whether a table with the name passed in

	TableName exists in the database.
ExplainPlan (inherited from TCustomIBCDataSet)	Used to obtain the query execution plan for Table, Query, and SQL components.
FetchAll	Defines whether to request all records of the query from database server when the dataset is being opened.
FetchRows (inherited from TCustomDADataset)	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL (inherited from TCustomDADataset)	Used to change the WHERE clause of SELECT statement and reopen a query.
FinalSQL (inherited from TCustomDADataset)	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
GeneratorMode (inherited from TCustomIBCDataSet)	Used to specify which method is used internally to generate a sequenced field.
GeneratorStep (inherited from TCustomIBCDataSet)	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
Handle (inherited from TCustomIBCDataSet)	Used to specify the handle for the SQL statement of TCustomIBCDataSet.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
IsQuery (inherited from TCustomIBCDataSet)	Used to check if the SQL statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
KeyFields (inherited from TCustomDADataset)	Used to build SQL statements for the

	SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.
KeyGenerator (inherited from TCustomIBCDataSet)	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
LockMode	Used to specify what kind of lock will be performed when editing a record.
MacroCount (inherited from TCustomDADataset)	Used to get the number of macros associated with the Macros property.
Macros (inherited from TCustomDADataset)	Makes it possible to change SQL queries easily.
MasterFields (inherited from TCustomDADataset)	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)	Used to specify the data source component which binds current dataset to the master one.
Options (inherited from TCustomIBCDataSet)	Used to specify the behaviour of the TCustomIBCDataSet object.
ParamCheck (inherited from TCustomDADataset)	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.

ParamCount (inherited from TCustomDADataset)	Used to indicate how many parameters are there in the Params property.
Params (inherited from TCustomDADataset)	Used to view and set parameter names, values, and data types dynamically.
Plan (inherited from TCustomIBCDataset)	Used to get or set the PLAN clause of the SELECT statement.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
ReadOnly (inherited from TCustomDADataset)	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions (inherited from TCustomDADataset)	Used to indicate when the editing record is refreshed.
RowsAffected (inherited from TCustomDADataset)	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
RowsDeleted (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were deleted during the last query operation.
RowsFetched (inherited from TCustomIBCDataset)	Used to get the number of the currently fetched rows.
RowsInserted (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were inserted during the last query operation.
RowsUpdated (inherited from TCustomIBCDataset)	Used to indicate the number of rows that were updated during the last query operation.
SmartFetch (inherited from TCustomIBCDataset)	The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.
SQL (inherited from TCustomDADataset)	Used to provide a SQL statement that a query

	component executes when its Open method is called.
SQLDelete (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying a deletion to a record.
SQLInsert (inherited from TCustomDADataset)	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
SQLLock (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to perform a record lock.
SQLRecCount (inherited from TCustomDADataset)	Used to specify the SQL statement that is used to get the record count when opening a dataset.
SQLRefresh (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.
SQLType (inherited from TCustomIBCDataSet)	Used to get the typecode of the SQL statement being processed by the InterBase database server.
SQLUpdate (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying an update to a dataset.
TableName	Used to specify the name of the database table this component encapsulates.
Transaction (inherited from TCustomIBCDataSet)	Used to determine the transaction under which the query of this dataset executes.
UniDirectional (inherited from TCustomDADataset)	Used if an application does not need bidirectional access to records in the result set.
UpdateObject (inherited from TCustomIBCDataSet)	Used to specify an update object component which provides SQL statements

	that perform updates of the read-only datasets.
UpdateRecordTypes (inherited from TMemDataSet)	Used to indicate the update status for the current record when cached updates are enabled.
UpdatesPending (inherited from TMemDataSet)	Used to check the status of the cached updates buffer.
UpdateTransaction (inherited from TCustomIBCDataset)	Used to get or set the transaction for modifying a dataset.

Methods

Name	Description
AddWhere (inherited from TCustomDADataset)	Adds condition to the WHERE clause of SELECT statement in the SQL property.
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
BreakExec (inherited from TCustomDADataset)	Breaks execution of the SQL statement on the server.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
CreateBlobStream (inherited from TCustomDADataset)	Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
CreateProcCall (inherited from TCustomIBCDataset)	Assigns PL/SQL block that calls stored procedure to the SQL property.

DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
DeleteTable (inherited from TCustomIBCTable)	Deletes a table from a database.
DeleteWhere (inherited from TCustomDADataset)	Removes WHERE clause from the SQL property and assigns the BaseSQL property.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
EmptyTable (inherited from TCustomIBCTable)	Truncates the current table content on the server.
Execute (inherited from TCustomDADataset)	Overloaded. Executes a SQL statement on the server.
Executing (inherited from TCustomDADataset)	Indicates whether SQL statement is still being executed.
Fetched (inherited from TCustomDADataset)	Used to find out whether TCustomDADataset has fetched all rows.
Fetching (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is still fetching rows.
FetchingAll (inherited from TCustomDADataset)	Used to learn whether TCustomDADataset is fetching all rows to the end.
FindKey (inherited from TCustomDADataset)	Searches for a record which contains specified field values.
FindMacro (inherited from TCustomDADataset)	Finds a macro with the specified name.
FindNearest (inherited from TCustomDADataset)	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.
FindParam (inherited from TCustomIBCDataSet)	Determines if a parameter with the specified name

	exists in a dataset.
GetArray (inherited from TCustomIBCDataSet)	Retrieves a TIBCArray object for a field when only its name is known.
GetBlob (inherited from TCustomIBCDataSet)	Retrieves a TIBCBlob object for a field when only its name is known.
GetDataType (inherited from TCustomDADDataSet)	Returns internal field types defined in the MemData and accompanying modules.
GetFieldObject (inherited from TCustomDADDataSet)	Returns a multireference shared object from field.
GetFieldPrecision (inherited from TCustomDADDataSet)	Retrieves the precision of a number field.
GetFieldScale (inherited from TCustomDADDataSet)	Retrieves the scale of a number field.
GetKeyFieldNames (inherited from TCustomDADDataSet)	Provides a list of available key field names.
GetOrderBy (inherited from TCustomDADDataSet)	Retrieves an ORDER BY clause from a SQL statement.
GotoCurrent (inherited from TCustomDADDataSet)	Sets the current record in this dataset similar to the current record in another dataset.
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.
Lock (inherited from TCustomDADDataSet)	Locks the current record.
MacroByName (inherited from TCustomDADDataSet)	Finds a macro with the specified name.
ParamByName (inherited from TCustomIBCDataSet)	Called to set or use parameter information for a specific parameter based on its name.

Prepare (inherited from TCustomDADataset)	Allocates, opens, and parses cursor for a query.
RefreshRecord (inherited from TCustomDADataset)	Actualizes field values for the current record.
RestoreSQL (inherited from TCustomDADataset)	Restores the SQL property modified by AddWhere and SetOrderBy.
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.
SaveSQL (inherited from TCustomDADataset)	Saves the SQL property value to BaseSQL.
SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
SetOrderBy (inherited from TCustomDADataset)	Builds an ORDER BY clause of a SELECT statement.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
SQLSaved (inherited from TCustomDADataset)	Determines if the SQL property value was saved to the BaseSQL property.
UnLock (inherited from TCustomDADataset)	Releases a record lock.
UnPrepare (inherited from TMemDataSet)	Frees the resources allocated for a previously prepared query on the server and client sides.

UpdateResult (inherited from TMemDataSet)	Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.
UpdateStatus (inherited from TMemDataSet)	Indicates the current update status for the dataset when cached updates are enabled.

Events

Name	Description
AfterExecute (inherited from TCustomDADataset)	Occurs after a component has executed a query to database.
AfterFetch (inherited from TCustomDADataset)	Occurs after dataset finishes fetching data from server.
AfterUpdateExecute (inherited from TCustomDADataset)	Occurs after executing insert, delete, update, lock and refresh operations.
BeforeFetch (inherited from TCustomDADataset)	Occurs before dataset is going to fetch block of records from the server.
BeforeUpdateExecute (inherited from TCustomDADataset)	Occurs before executing insert, delete, update, lock, and refresh operations.
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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.19.2 Properties

Properties of the **TIBCTable** class.

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

Public

Name	Description
AutoCommit (inherited from TCustomIBCDataSet)	Used to automatically commit each update, insert or delete statement by database server.
BaseSQL (inherited from TCustomDADDataSet)	Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
Conditions (inherited from TCustomDADDataSet)	Used to add WHERE conditions to a query
Connection (inherited from TCustomIBCDataSet)	Used to specify the connection in which the dataset will be executed.
Cursor (inherited from TCustomIBCDataSet)	Used for positioned UPDATE and DELETE statements made for the data retrieved with the SELECT statements with the FOR UPDATE clause.
DataTypeMap (inherited from TCustomDADDataSet)	Used to set data type mapping rules
Debug (inherited from TCustomDADDataSet)	Used to display the statement that is being executed and the values and types of its parameters.
DetailFields (inherited from TCustomDADDataSet)	Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.
Disconnected (inherited from TCustomDADDataSet)	Used to keep dataset opened after connection is closed.
DMLRefresh (inherited from TCustomIBCDataSet)	Used to refresh record by the RETURNING clause when insert is performed.
Encryption (inherited from TCustomIBCDataSet)	Used to specify encryption options in a dataset.

Exists (inherited from TCustomIBCTable)	Indicates whether a table with the name passed in TableName exists in the database.
ExplainPlan (inherited from TCustomIBCDataSet)	Used to obtain the query execution plan for Table, Query, and SQL components.
FetchRows (inherited from TCustomDADDataSet)	Used to define the number of rows to be transferred across the network at the same time.
FilterSQL (inherited from TCustomDADDataSet)	Used to change the WHERE clause of SELECT statement and reopen a query.
FinalSQL (inherited from TCustomDADDataSet)	Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.
GeneratorMode (inherited from TCustomIBCDataSet)	Used to specify which method is used internally to generate a sequenced field.
GeneratorStep (inherited from TCustomIBCDataSet)	Used to set the increment for increasing or decreasing current generator value when using the automatic key field value generation feature.
Handle (inherited from TCustomIBCDataSet)	Used to specify the handle for the SQL statement of TCustomIBCDataSet.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
IsQuery (inherited from TCustomIBCDataSet)	Used to check if the SQL statement returns rows.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
KeyFields (inherited from TCustomDADDataSet)	Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before

	updating the database.
KeyGenerator (inherited from TCustomIBCDataSet)	Used to specify the name of a generator that will be used to fill in a key field after a new record is inserted or posted to the database.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
MacroCount (inherited from TCustomDADDataSet)	Used to get the number of macros associated with the Macros property.
Macros (inherited from TCustomDADDataSet)	Makes it possible to change SQL queries easily.
MasterFields (inherited from TCustomDADDataSet)	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 TCustomDADDataSet)	Used to specify the data source component which binds current dataset to the master one.
Options (inherited from TCustomIBCDataSet)	Used to specify the behaviour of the TCustomIBCDataSet object.
ParamCheck (inherited from TCustomDADDataSet)	Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.
ParamCount (inherited from TCustomDADDataSet)	Used to indicate how many parameters are there in the Params property.
Params (inherited from TCustomDADDataSet)	Used to view and set parameter names, values, and data types dynamically.

Plan (inherited from TCustomIBCDataSet)	Used to get or set the PLAN clause of the SELECT statement.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
ReadOnly (inherited from TCustomDADataset)	Used to prevent users from updating, inserting, or deleting data in the dataset.
RefreshOptions (inherited from TCustomDADataset)	Used to indicate when the editing record is refreshed.
RowsAffected (inherited from TCustomDADataset)	Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.
RowsDeleted (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were deleted during the last query operation.
RowsFetched (inherited from TCustomIBCDataSet)	Used to get the number of the currently fetched rows.
RowsInserted (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were inserted during the last query operation.
RowsUpdated (inherited from TCustomIBCDataSet)	Used to indicate the number of rows that were updated during the last query operation.
SmartFetch (inherited from TCustomIBCDataSet)	The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.
SQL (inherited from TCustomDADataset)	Used to provide a SQL statement that a query component executes when its Open method is called.
SQLDelete (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying a deletion to a record.

SQLInsert (inherited from TCustomDADataset)	Used to specify the SQL statement that will be used when applying an insertion to a dataset.
SQLLock (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to perform a record lock.
SQLRecCount (inherited from TCustomDADataset)	Used to specify the SQL statement that is used to get the record count when opening a dataset.
SQLRefresh (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.
SQLType (inherited from TCustomIBCDataSet)	Used to get the typecode of the SQL statement being processed by the InterBase database server.
SQLUpdate (inherited from TCustomDADataset)	Used to specify a SQL statement that will be used when applying an update to a dataset.
Transaction (inherited from TCustomIBCDataSet)	Used to determine the transaction under which the query of this dataset executes.
UniDirectional (inherited from TCustomDADataset)	Used if an application does not need bidirectional access to records in the result set.
UpdateObject (inherited from TCustomIBCDataSet)	Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.
UpdateRecordTypes (inherited from TMemDataSet)	Used to indicate the update status for the current record when cached updates are enabled.
UpdatesPending (inherited from TMemDataSet)	Used to check the status of the cached updates buffer.

UpdateTransaction (inherited from TCustomIBDataSet)	Used to get or set the transaction for modifying a dataset.
--	---

Published

Name	Description
FetchAll	Defines whether to request all records of the query from database server when the dataset is being opened.
LockMode	Used to specify what kind of lock will be performed when editing a record.
TableName	Used to specify the name of the database table this component encapsulates.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.19.2.1 FetchAll Property

Defines whether to request all records of the query from database server when the dataset is being opened.

Class

[TIBCTable](#)

Syntax

```
property FetchAll: boolean;
```

Remarks

When set to True, all records of the query are requested from database server when the

dataset is being opened. When set to False, records are retrieved when a data-aware component or a program requests it. If a query can return a lot of records, set this property to False if initial response time is important.

When the FetchAll property is False, the first call to [TMemDataSet.Locate](#) and [TMemDataSet.LocateEx](#) methods may take a lot of time to retrieve additional records to the client side.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.19.2.2 LockMode Property

Used to specify what kind of lock will be performed when editing a record.

Class

[TIBCTable](#)

Syntax

```
property LockMode: TLockMode default ImNone;
```

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

- [TIBCStoredProc.LockMode](#)
- [TIBCQuery.LockMode](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.19.2.3 TableName Property

Used to specify the name of the database table this component encapsulates.

Class

[TIBCTable](#)

Syntax

```
property TableName: string;
```

Remarks

Use the TableName property to specify the name of the database table this component encapsulates. If [TCustomDADataset.Connection](#) is assigned at design time and correct connection settings are provided, select a valid table name from the TableName drop-down list in Object Inspector.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20 TIBCTransaction Class

A component for managing transactions in an application.

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

Unit

[IBC](#)

Syntax

```
TIBCTransaction = class(TDATransaction);
```

Remarks

The TIBCTransaction component is used to provide discrete transaction control over connection. It can be used for manipulating simple local and global transactions.

All components which are dedicated to perform data access, such as TIBCQuery, TIBCSQL, TIBCScript, must have their Transaction property assigned with one of TIBCTransaction instances.

Inheritance Hierarchy

[TDATransaction](#)

TIBCTransaction

See Also

- [TCustomDACConnection.StartTransaction](#)
- [TCustomDACConnection.Commit](#)
- [TCustomDACConnection.Rollback](#)
- [TIBCCConnection.DefaultTransaction](#)
- [TIBCCConnection.Transactions](#)
- [TCustomIBCDataSet.Transaction](#)
- [TIBCSQL.Transaction](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.1 Members

[TIBCTransaction](#) class overview.

Properties

Name	Description
Active	Determines if the transaction is active or not.
Connections	Used to specify a connection for the given index.
ConnectionsCount	Used to get the number of connections associated with the transaction component.
DefaultCloseAction	Used to specify the transaction behaviour when connection closes.
DefaultConnection	Used to access the default connection of the transaction.

Handle	Used to specify the handle of the transaction.
IsolationLevel	Used to get or set the transaction isolation level and access mode.
Params	Used to access transaction parameters of the transaction parameter buffer.

Methods

Name	Description
AddConnection	Binds a TCustomDAConnection object with the transaction component.
Commit	Stores all changes of data associated with the transaction to the database server permanently.
CommitRetaining	Stores to the database server all changes of data associated with the transaction permanently and then retains the transaction context.
FindDefaultConnection	Returns the default connection for the transaction.
ReleaseSavepoint	Destroys the specified savepoint without affecting any work that has been performed after its creation.
RemoveConnection	Disassociates the specified connections from the transaction.
Rollback	Rolls back all changes of data associated with the transaction.
RollbackRetaining	Rolls back all data changes associated with the transaction and retains the transaction context.

RollbackSavepoint	Cancels all updates for the current transaction and restores its state up to the moment of the last defined savepoint.
StartSavepoint	Defines a point in the transaction to which you can roll back later.
StartTransaction (inherited from TDATransaction)	Begins a new transaction.

Events

Name	Description
OnCommit (inherited from TDATransaction)	Occurs after the transaction has been successfully committed.
OnCommitRetaining (inherited from TDATransaction)	Occurs after CommitRetaining has been executed.
OnError	Occurs when processing errors that are raised during executing transaction and savepoint control statements such as COMMIT, ROLLBACK, SAVEPOINT, RELEASE SAVEPOINT and others.
OnRollback (inherited from TDATransaction)	Occurs after the transaction has been successfully rolled back.
OnRollbackRetaining (inherited from TDATransaction)	Occurs after RollbackRetaining has been executed.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.2 Properties

Properties of the **TIBCTransaction** class.

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

Public

Name	Description
Connections	Used to specify a connection for the given index.
ConnectionsCount	Used to get the number of connections associated with the transaction component.
Handle	Used to specify the handle of the transaction.

Published

Name	Description
Active	Determines if the transaction is active or not.
DefaultCloseAction	Used to specify the transaction behaviour when connection closes.
DefaultConnection	Used to access the default connection of the transaction.
IsolationLevel	Used to get or set the transaction isolation level and access mode.
Params	Used to access transaction parameters of the transaction parameter buffer.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.11.1.20.2.1 Active Property

Determines if the transaction is active or not.

Class

[TIBCTransaction](#)

Syntax

```
property Active: Boolean stored IsActiveStored default False;
```

Remarks

The Active property is used to indicate whether transaction is active or not.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.2.2 Connections Property(Indexer)

Used to specify a connection for the given index.

Class

[TIBCTransaction](#)

Syntax

```
property Connections[Index: integer]: TIBConnection;
```

Parameters

Index

Holds the index for which to specify a connection.

Remarks

Use the Connections property to specify a connection for the given index.

See Also

- [ConnectionsCount](#)
- [AddConnection](#)
- [RemoveConnection](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.2.3 ConnectionsCount Property

Used to get the number of connections associated with the transaction component.

Class

[TIBCTransaction](#)

Syntax

```
property ConnectionsCount: integer;
```

Remarks

Use the ConnectionsCount property for getting the number of connections associated with the transaction component.

See Also

- [AddConnection](#)
- [RemoveConnection](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.2.4 DefaultCloseAction Property

Used to specify the transaction behaviour when connection closes.

Class

[TIBCTransaction](#)

Syntax

```
property DefaultCloseAction: TIBCTransactionAction default  
taRollback;
```

Remarks

Use the DefaultCloseAction property to specify the transaction behaviour when connection closes.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.2.5 DefaultConnection Property

Used to access the default connection of the transaction.

Class

[TIBCTransaction](#)

Syntax

```
property DefaultConnection: TIBConnection stored  
IsInternalTrStored;
```

Remarks

Use the DefaultConnection property to access the default connection of the transaction.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.2.6 Handle Property

Used to specify the handle of the transaction.

Class

[TIBCTransaction](#)

Syntax

```
property Handle: TISC_TR_HANDLE;
```

Remarks

Use the Handle property to specify the handle of the transaction. Use Handle Property to make calls directly to the InterBase API. Some of the InterBase API functions require a transaction handle as an argument.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.2.7 IsolationLevel Property

Used to get or set the transaction isolation level and access mode.

Class

[TIBCTransaction](#)

Syntax

```
property IsolationLevel: TIBCIsoationLevel default  
iblReadCommitted;
```

Remarks

Use the IsolationLevel property to get or set the transaction isolation level and access mode. You should add IBCClasses unit to the uses list to use this property.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.2.8 Params Property

Used to access transaction parameters of the transaction parameter buffer.

Class

[TIBCTransaction](#)

Syntax

```
property Params: TStrings;
```

Remarks

Use the Params property to access transaction parameters of the transaction parameter buffer. Refer to InterBase API Guide for more information on this parameters.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.3 Methods

Methods of the **TIBCTransaction** class.

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

Public

Name	Description
AddConnection	Binds a TCustomDAConnection object with the transaction component.
Commit	Stores all changes of data associated with the transaction to the database server permanently.
CommitRetaining	Stores to the database server all changes of data associated with the transaction permanently and then retains the transaction context.
FindDefaultConnection	Returns the default connection for the transaction.
ReleaseSavepoint	Destroys the specified savepoint without affecting any work that has been performed after its creation.
RemoveConnection	Disassociates the specified connections from the transaction.
Rollback	Rolls back all changes of data associated with the transaction.
RollbackRetaining	Rolls back all data changes associated with the transaction and retains the transaction context.
RollbackSavepoint	Cancels all updates for the current transaction and restores its state up to the moment of the last defined

	savepoint.
StartSavepoint	Defines a point in the transaction to which you can roll back later.
StartTransaction (inherited from TDATransaction)	Begins a new transaction.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.3.1 AddConnection Method

Binds a TCustomDACConnection object with the transaction component.

Class

[TIBCTransaction](#)

Syntax

```
function AddConnection(Connection: TIBConnection): integer;
```

Parameters

Connection

Holds a TCustomDACConnection object to associate with the transaction component.

Return Value

the index of associated connection in the connection list.

Remarks

Use the AddConnection method to associate a TCustomDACConnection object with the transaction component.

See Also

- [RemoveConnection](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.11.1.20.3.2 Commit Method

Stores all changes of data associated with the transaction to the database server permanently.

Class

[TIBCTransaction](#)

Syntax

```
procedure Commit; override;
```

Remarks

Call the Commit method to store to the database server all changes of data associated with the transaction permanently.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.3.3 CommitRetaining Method

Stores to the database server all changes of data associated with the transaction permanently and then retains the transaction context.

Class

[TIBCTransaction](#)

Syntax

```
procedure CommitRetaining;
```

Remarks

Call the CommitRetaining method to store to the database server all changes of data associated with the transaction permanently and then retain the transaction context.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.3.4 FindDefaultConnection Method

Returns the default connection for the transaction.

Class

[TIBCTransaction](#)

Syntax

```
function FindDefaultConnection: TIBConnection;
```

Remarks

Call the FindDefaultConnection method to return the default connection for the transaction.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.3.5 ReleaseSavepoint Method

Destroys the specified savepoint without affecting any work that has been performed after its creation.

Class

[TIBCTransaction](#)

Syntax

```
procedure ReleaseSavepoint(const Name: string);
```

Parameters

Name

Holds the savepoint name.

Remarks

Call the ReleaseSavepoint method to destroy the specified savepoint without affecting any work that has been performed after its creation.

See Also

- [StartSavepoint](#)

- [RollbackSavepoint](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.3.6 RemoveConnection Method

Disassociates the specified connections from the transaction.

Class

[TIBCTransaction](#)

Syntax

```
procedure RemoveConnection(Connection: TIBConnection);
```

Parameters

Connection

Holds the connections to disassociate.

Remarks

Use the RemoveConnection method to disassociate the specified connections from the transaction.

See Also

- [AddConnection](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.3.7 Rollback Method

Rolls back all changes of data associated with the transaction.

Class

[TIBCTransaction](#)

Syntax

```
procedure Rollback; override;
```

Remarks

Call the Rollback method to roll back all changes of data associated with the transaction.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.3.8 RollbackRetaining Method

Rolls back all data changes associated with the transaction and retains the transaction context.

Class

[TIBCTransaction](#)

Syntax

```
procedure RollbackRetaining;
```

Remarks

Call the RollbackRetaining method to roll back all changes of data associated with the transaction and retain the transaction context.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.3.9 RollbackSavepoint Method

Cancels all updates for the current transaction and restores its state up to the moment of the last defined savepoint.

Class

[TIBCTransaction](#)

Syntax

```
procedure RollbackSavepoint(const Name: string);
```

Parameters

Name

Holds the defined savepoint name.

Remarks

Call the RollbackSavepoint to cancel all updates for the current transaction and restore its state up to the moment of the last defined savepoint.

See Also

- [Rollback](#)
- [StartSavepoint](#)
- [ReleaseSavepoint](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.20.3.10 StartSavepoint Method

Defines a point in the transaction to which you can roll back later.

Class

[TIBCTransaction](#)

Syntax

```
procedure StartSavepoint(const Name: string);
```

Parameters

Name

Holds the savepoint name.

Remarks

Call the StartSavepoint method to define a point in the transaction to which you can roll back later. As the parameter, you can pass any valid name to identify the savepoint.

To roll back to the last savepoint call [RollbackSavepoint](#).

See Also

- [RollbackSavepoint](#)
- [ReleaseSavepoint](#)

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.11.1.20.4 Events

Events of the **TIBCTransaction** class.

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

Public

Name	Description
OnCommit (inherited from TDATransaction)	Occurs after the transaction has been successfully committed.
OnCommitRetaining (inherited from TDATransaction)	Occurs after CommitRetaining has been executed.
OnRollback (inherited from TDATransaction)	Occurs after the transaction has been successfully rolled back.
OnRollbackRetaining (inherited from TDATransaction)	Occurs after RollbackRetaining has been executed.

Published

Name	Description
OnError	Occurs when processing errors that are raised during executing transaction and savepoint control statements such as COMMIT, ROLLBACK, SAVEPOINT, RELEASE SAVEPOINT and others.

See Also

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

Devart. All Rights Reserved.

5.11.1.20.4.1 OnError Event

Occurs when processing errors that are raised during executing transaction and savepoint control statements such as COMMIT, ROLLBACK, SAVEPOINT, RELEASE SAVEPOINT and others.

Class

[TIBCTransaction](#)

Syntax

```
property OnError: TIBCTransactionErrorEvent;
```

Remarks

Write the OnError event handler to process errors that occur during executing transaction and savepoint control statements such as COMMIT, ROLLBACK, SAVEPOINT, RELEASE SAVEPOINT and others. Check the E parameter to get an error code.

Note: You should explicitly add T:Devart.IbDac.Units.IBCError unit to 'uses' list to use OnError event handler.

This event occur only when two or more connections are associated with the transaction. When only one connection is assigned to the transaction, then the OnError event of the TIBConnection class arises.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.21 TIBCUpdateSQL Class

A component for tuning update operations for the DataSet component.

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

Unit

[IBC](#)

Syntax

```
TIBCUpdateSQL = class(TCustomDAUpdateSQL);
```

Remarks

Use the TIBCUpdateSQL component to provide DML statements for the dataset components that return read-only result set. This component also allows setting objects that can be used for executing update operations. You may prefer to use directly SQLInsert, SQLUpdate, and SQLDelete properties of the [TCustomDADataSet](#) descendants.

Inheritance Hierarchy

[TCustomDAUpdateSQL](#)

TIBCUpdateSQL

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.1.21.1 Members

[TIBCUpdateSQL](#) class overview.

Properties

Name	Description
DataSet (inherited from TCustomDAUpdateSQL)	Used to hold a reference to the TCustomDADataSet object that is being updated.
DeleteObject (inherited from TCustomDAUpdateSQL)	Provides ability to perform advanced adjustment of the delete operations.
DeleteSQL (inherited from TCustomDAUpdateSQL)	Used when deleting a record.
InsertObject (inherited from TCustomDAUpdateSQL)	Provides ability to perform advanced adjustment of insert operations.
InsertSQL (inherited from TCustomDAUpdateSQL)	Used when inserting a record.
LockObject (inherited from TCustomDAUpdateSQL)	Provides ability to perform advanced adjustment of lock operations.
LockSQL (inherited from TCustomDAUpdateSQL)	Used to lock the current record.

ModifyObject (inherited from TCustomDAUpdateSQL)	Provides ability to perform advanced adjustment of modify operations.
ModifySQL (inherited from TCustomDAUpdateSQL)	Used when updating a record.
RefreshObject (inherited from TCustomDAUpdateSQL)	Provides ability to perform advanced adjustment of refresh operations.
RefreshSQL (inherited from TCustomDAUpdateSQL)	Used to specify an SQL statement that will be used for refreshing the current record by TCustomDADataset.RefreshRecord procedure.
SQL (inherited from TCustomDAUpdateSQL)	Used to return a SQL statement for one of the ModifySQL, InsertSQL, or DeleteSQL properties.

Methods

Name	Description
Apply (inherited from TCustomDAUpdateSQL)	Sets parameters for a SQL statement and executes it to update a record.
ExecSQL (inherited from TCustomDAUpdateSQL)	Executes a SQL statement.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.2 Types

Types in the **IBC** unit.

Types

Name	Description
TIBCTransactionErrorEvent	This type is used for the TIBCTransaction.OnError event.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.11.2.1 TIBCTransactionErrorEvent Procedure Reference

This type is used for the [TIBCTransaction.OnError](#) event.

Unit

[IBC](#)

Syntax

```
TIBCTransactionErrorEvent = procedure (Sender: TObject; E:
EIBCErrror) of object;
```

Parameters

Sender

An object that raised the event.

E

Holds the error code.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.3 Enumerations

Enumerations in the **IBC** unit.

Enumerations

Name	Description
TGeneratorMode	Specifies the method used internally to generate a sequenced field.
TIBCTProtocol	Specifies the network protocol of connection with InterBase server.
TIBCTransactionAction	Specifies the transaction behaviour when connection closes.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.3.1 TGeneratorMode Enumeration

Specifies the method used internally to generate a sequenced field.

Unit

[IBC](#)

Syntax

```
TGeneratorMode = (gmInsert, gmPost);
```

Values

Value	Meaning
gmInsert	New record is inserted into the dataset where the first key field populated with a sequenced value. Application may modify this field before posting the record to the database.
gmPost	Database server populates the key field with a sequenced value when application posts the record to the database. However if user specified the key field value, the key generator will not be used.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.3.2 TIBCProtocol Enumeration

Specifies the network protocol of connection with InterBase server.

Unit

[IBC](#)

Syntax

```
TIBCProtocol = (TCP, NetBEUI, SPX);
```

Values

Value	Meaning
NetBEUI	Uses the NetBEUI protocol.
SPX	Uses the SPX protocol.
TCP	Uses the TCP protocol. The default value.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.3.3 TIBCTransactionAction Enumeration

Specifies the transaction behaviour when connection closes.

Unit

[IBC](#)

Syntax

```
TIBCTransactionAction = (taCommit, taRollback, taCommitRetaining, taRollbackRetaining);
```

Values

Value	Meaning
taCommit	Transaction is committed and is closed.
taCommitRetaining	Transaction is committed and remains opened.
taRollback	Transaction is rolled back and is closed.
taRollbackRetaining	Transaction is rolled back and remains opened.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.4 Variables

Variables in the **IBC** unit.

Variables

Name	Description
Connections	Holds pointers to all TIBCCConnection objects of an application.
DefConnection	Read this variable to get pointer to default connection object. Same as DefaultConnection function.

UseDefConnection	When set to true enables TCustomIBCDataSet and TIBCSQL components to use default connection if they are not attached to any connection.
----------------------------------	---

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.4.1 Connections Variable

Holds pointers to all TIBCConnection objects of an application.

Unit

[IBC](#)

Syntax

```
Connections: TConnectionList;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.4.2 DefConnection Variable

Read this variable to get pointer to default connection object. Same as DefaultConnection function.

Unit

[IBC](#)

Syntax

```
DefConnection: TIBCConnection;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.4.3 UseDefConnection Variable

When set to true enables TCustomIBCDataset and TIBCSQL components to use default connection if they are not attached to any connection.

Unit

[IBC](#)

Syntax

```
UseDefConnection: boolean;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.5 Constants

Constants in the **IBC** unit.

Constants

Name	Description
IBDACVersion	Read this constant to get current version number for IBDAC.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.11.5.1 IBDACVersion Constant

Read this constant to get current version number for IBDAC.

Unit

[IBC](#)

Syntax

```
IBDACVersion = '9.1.0';
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12 IBCAdmin

This unit contains implementation of components, used for InterBase/Firebird server administration.

Classes

Name	Description
TCustomIBCSERVICE	TCustomIBCSERVICE is the ancestor object from which all TIBCSERVICE components descend.
TIBCSERVICEBackupRestoreService	TIBCSERVICEBackupRestoreService is the base class from which TIBCSERVICEBackupService and TIBCSERVICERestoreService are derived.
TIBCSERVICEBackupService	Used to backup a database.
TIBCSERVICEConfigParams	TIBCSERVICEConfigParams holds the configuration information about an InterBase server.
TIBCSERVICEConfigService	Use a TIBCSERVICEConfigService object to configure database parameters.
TIBCSERVICEControlAndQueryService	TIBCSERVICEControlAndQueryService is the base class from which the log, statistical, validation, security, and backup and restore TIBCSERVICE components descend.
TIBCSERVICEDatabaseInfo	Describes an InterBase database.
TIBCSERVICEJournalInformation	A class used to access the TIBCSERVICEConfigService component properties.
TIBCSERVICELicenseInfo	Stores information about licensed users.
TIBCSERVICELicenseMaskInfo	Indicates the software activation certificate options enabled on the server.
TIBCSERVICELicensingService	TIBCSERVICELicensingService configures the licensing parameters

TIBCLimboTransactionInfo	TIBCLimboTransactionInfo stores information about a limbo transaction.
TIBCLogService	Returns the contents of the interbase.log file from server.
TIBCRestoreService	Used to restore a database.
TIBCSecurityService	Used to manage user access to the InterBase server.
TIBCServerProperties	A class that returns database server information.
TIBCStatisticalService	TIBCStatisticalService is used to view database statistics.
TIBCTraceService	This component is used for working with trace service added in Firebird 2.5.
TIBCUserInfo	TIBCUserInfo stores information about an InterBase user for the security service.
TIBCValidationService	Used to validate a database and reconcile database transactions.
TIBCVersionInfo	Represents the version information about an InterBase server.

Types

Name	Description
TIBCBackupOptions	Represents the set of TIBCBackupOption .
TIBCNBackupOptions	Represents the set of TIBCNBackupOption .
TIBCRestoreOptions	Represents the set of TIBCRestoreOption .
TIBCStatOptions	Represents the set of TIBCStatOption .
TIBCValidateOptions	Represents the set of TIBCValidateOption .

Enumerations

Name	Description
TIBCBackupOption	Allows you to build backup options into your application.
TIBCLicensingAction	Allows to add or remove an InterBase software activation certificate.
TIBCNBackupOption	Options used in TIBCBackupRestoreService for nBackup.
TIBCRestoreOption	Specifies the data restore parameters.
TIBCSecurityAction	Specify the type of the operation for InterBase Security Service to perform.
TIBCStatOption	Allows to specify the way the database statistics would be requested.
TIBCTransactionAdvise	Allows to specify the suggested action for ending the transaction.
TIBCTransactionGlobalAction	Determines which action to take concerning limbo transactions.
TIBCTransactionState	Allows to specify the current state of the limbo transaction.
TIBCValidateOption	Sets the options of validation.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1 Classes

Classes in the **IBCAAdmin** unit.

Classes

Name	Description
TCustomIBCSERVICE	TCustomIBCSERVICE is the

	ancestor object from which all TIBCSERVICE components descend.
TIBCBACKUPRESTORESERVICE	TIBCBACKUPRESTORESERVICE is the base class from which TIBCBACKUPSERVICE and TIBCRESTORESERVICE are derived.
TIBCBACKUPSERVICE	Used to backup a database.
TIBCCONFIGPARAMS	TIBCCONFIGPARAMS holds the configuration information about an InterBase server.
TIBCCONFIGSERVICE	Use a TIBCCONFIGSERVICE object to configure database parameters.
TIBCCONTROLANDQUERYSERVICE	TIBCCONTROLANDQUERYSERVICE is the base class from which the log, statistical, validation, security, and backup and restore TIBCSERVICE components descend.
TIBCDATABASEINFO	Describes an InterBase database.
TIBCJOURNALINFORMATION	A class used to access the TIBCCONFIGSERVICE component properties.
TIBCLICENSEINFO	Stores information about licensed users.
TIBCLICENSEMASKINFO	Indicates the software activation certificate options enabled on the server.
TIBCLICENSINGSERVICE	TIBCLICENSINGSERVICE configures the licensing parameters
TIBCLIMBOTRANSACTIONINFO	TIBCLIMBOTRANSACTIONINFO stores information about a limbo transaction.
TIBCLOGSERVICE	Returns the contents of the interbase.log file from server.
TIBCRESTORESERVICE	Used to restore a database.

TIBCSecurityService	Used to manage user access to the InterBase server.
TIBCServerProperties	A class that returns database server information.
TIBCStatisticalService	TIBCStatisticalService is used to view database statistics.
TIBCTraceService	This component is used for working with trace service added in Firebird 2.5.
TIBCUserInfo	TIBCUserInfo stores information about an InterBase user for the security service.
TIBCValidationService	Used to validate a database and reconcile database transactions.
TIBCVersionInfo	Represents the version information about an InterBase server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1 TCustomIBCSERVICE Class

TCustomIBCSERVICE is the ancestor object from which all TIBCSERVICE components descend.

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

Unit

[IBCAAdmin](#)

Syntax

```
TCustomIBCSERVICE = class(TComponent);
```

Remarks

TCustomIBCSERVICE is the ancestor object from which all TIBCSERVICE components descend.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.12.1.1.1 Members

[TCustomIBService](#) class overview.

Properties

Name	Description
Active	Used to set the service to active or inactive (default).
Handle	Used to return the database handle.
LoginPrompt	Used to display a login prompt before attaching to a database.
Params	Used to set or return database parameters.
Protocol	Used to select the network protocol.
Server	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB	Used to return and sets SPB parameters.

Methods

Name	Description
Attach	Attaches to the database.
Detach	Detaches from the database.
ServiceStart	Starts the service.

Events

Name	Description
OnAttach	Occurs when the database is attached.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.2 Properties

Properties of the **TCustomIBCSERVICE** class.

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

Public

Name	Description
Handle	Used to return the database handle.
ServiceParamBySPB	Used to return and sets SPB parameters.

Published

Name	Description
Active	Used to set the service to active or inactive (default).
LoginPrompt	Used to display a login prompt before attaching to a database.
Params	Used to set or return database parameters.
Protocol	Used to select the network protocol.
Server	Used to set the name of the server on which the services are to be run.

See Also

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

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.2.1 Active Property

Used to set the service to active or inactive (default).

Class

[TCustomIBCSERVICE](#)

Syntax

```
property Active: Boolean default False;
```

Remarks

Use the Active property to set the service to active (True) or the default, inactive (False).

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.2.2 Handle Property

Used to return the database handle.

Class

[TCustomIBCSERVICE](#)

Syntax

```
property Handle: TISC_SVC_HANDLE;
```

Remarks

Use the Handle property to return the database handle.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.2.3 LoginPrompt Property

Used to display a login prompt before attaching to a database.

Class

[TCustomIBCSERVICE](#)

Syntax

```
property LoginPrompt: Boolean default DefValLoginPrompt;
```

Remarks

Use the LoginPrompt property to display (or not display) a login prompt before attaching to a database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.2.4 Params Property

Used to set or return database parameters.

Class

[TCustomIBCService](#)

Syntax

```
property Params: TStrings;
```

Remarks

Use the Params property to set or return database parameters.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.2.5 Protocol Property

Used to select the network protocol.

Class

[TCustomIBCService](#)

Syntax

```
property Protocol: TIBCSProtocol default DefValProtocol;
```

Remarks

Use the Protocol property to select the network protocol.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.2.6 Server Property

Used to set the name of the server on which the services are to be run.

Class

[TCustomIBCService](#)

Syntax

```
property Server: string;
```

Remarks

Use the Server property to set the name of the server on which the services are to be run.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.2.7 ServiceParamBySPB Property(Indexer)

Used to return and sets SPB parameters.

Class

[TCustomIBCService](#)

Syntax

```
property ServiceParamBySPB[const Idx: Integer]: string;
```

Parameters

Idx

Holds the index of parameter.

Remarks

Use the ServiceParamBySPB property to inspect and set SPB parameters.

For example, DBParamBySPB[isc_SPB_user_name] sets and inspects the user name.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.3 Methods

Methods of the **TCustomIBCSERVICE** class.

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

Public

Name	Description
Attach	Attaches to the database.
Detach	Detaches from the database.
ServiceStart	Starts the service.

See Also

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

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.3.1 Attach Method

Attaches to the database.

Class

[TCustomIBCSERVICE](#)

Syntax

```
procedure Attach;
```

Remarks

Call the Attach method to attach to the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.3.2 Detach Method

Detaches from the database.

Class

[TCustomIBCSERVICE](#)

Syntax

```
procedure Detach;
```

Remarks

Call the Detach method to detach from the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.3.3 ServiceStart Method

Starts the service.

Class

[TCustomIBCSERVICE](#)

Syntax

```
procedure ServiceStart; virtual;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.4 Events

Events of the **TCustomIBCSERVICE** class.

For a complete list of the **TCustomIBCSERVICE** class members, see the

[TCustomIBCSERVICE Members](#) topic.

Published

Name	Description
OnAttach	Occurs when the database is attached.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.1.4.1 OnAttach Event

Occurs when the database is attached.

Class

[TCustomIBCSERVICE](#)

Syntax

```
property OnAttach: TNotifyEvent;
```

Remarks

Write an OnAttach event handler to take specific actions when a database is attached. If an exception is raised in this event, the database is not attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.2 TIBCBACKUPRESTORESERVICE Class

TIBCBACKUPRESTORESERVICE is the base class from which [TIBCBACKUPSERVICE](#) and [TIBCRESTORESERVICE](#) are derived.

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

Unit

[IBCAAdmin](#)

Syntax

```
TIBCBackupRestoreService = class(TIBCControlAndQueryService);
```

Remarks

TIBCBackupRestoreService is the base class from which [TIBCBackupService](#) and [TIBCRestoreService](#) are derived.

Inheritance Hierarchy

[TCustomIBCSERVICE](#)[TIBCControlAndQueryService](#)**TIBCBackupRestoreService**

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.12.1.2.1 Members

[TIBCBackupRestoreService](#) class overview.

Properties

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BackupFile	Holds the path of the backup file name.
BufferSize (inherited from TIBCControlAndQueryService)	Used to set or return the buffer size.
Eof (inherited from TIBCControlAndQueryService)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.

NBackupLevel	Used to set backup level for nBackup.
NBackupOptions	Used to set backup options for nBackup.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.
UseNBackup	Used to enable or disable using nBackup service.
Verbose	Used to set or return the backup or restore in the verbose mode.

Methods

Name	Description
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
GetNextChunk (inherited from TIBCControlAndQueryService)	Returns the next chunk of data.
GetNextLine (inherited from TIBCControlAndQueryService)	Returns the next line of data.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.

Events

Name	Description
OnAttach (inherited from TCustomIBCSERVICE)	Occurs when the database is attached.

Devart. All Rights Reserved.

5.12.1.2.2 Properties

Properties of the **TIBCBackupRestoreService** class.

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

Public

Name	Description
BackupFile	Holds the path of the backup file name.
Eof (inherited from TIBCControlAndQueryService)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
NBackupLevel	Used to set backup level for nBackup.
NBackupOptions	Used to set backup options for nBackup.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.
UseNBackup	Used to enable or disable using nBackup service.
Verbose	Used to set or return the backup or restore in the verbose mode.

Published

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BufferSize (inherited from TIBCControlAndQueryService)	Used to set or return the buffer size.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a

	database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.2.2.1 BackupFile Property

Holds the path of the backup file name.

Class

[TIBCBackupRestoreService](#)

Syntax

```
property BackupFile: TStrings;
```

Remarks

The BackupFile property holds the path of the backup file name.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.2.2.2 NBackupLevel Property

Used to set backup level for nBackup.

Class

[TIBCBackupRestoreService](#)

Syntax

```
property NBackupLevel: integer default 0;
```

Remarks

Use the NBackupLevel property to set backup level for nBackup. This property is used only when UseNBackup = True.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.2.2.3 NBackupOptions Property

Used to set backup options for nBackup.

Class

[TIBCBackupRestoreService](#)

Syntax

```
property NBackupOptions: TIBCBackupOptions default [];
```

Remarks

Use the NBackupOptions property to set backup options for nBackup. This property is used only when UseNBackup = True.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.2.2.4 UseNBackup Property

Used to enable or disable using nBackup service.

Class

[TIBCBackupRestoreService](#)

Syntax

```
property UseNBackup: boolean default False;
```

Remarks

Use the UseNBackup property to enable or disable using nBackup service.

Set this property to True to use nBackup service instead of the standard backup/restore service.

Note: nBackup service is supported starting with Firebird 2.5.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.2.2.5 Verbose Property

Used to set or return the backup or restore in the verbose mode.

Class

[TIBCBackupRestoreService](#)

Syntax

```
property verbose: Boolean default False;
```

Remarks

Use the Verbose property to set or return the backup or restore in the verbose mode.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.3 TIBCBackupService Class

Used to backup a database.

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

Unit

[IBAdmin](#)

Syntax

```
TIBCBackupService = class(TIBCBackupRestoreService);
```

Remarks

Use a TIBCBackupService object to backup a database.

Inheritance Hierarchy

[TCustomIBCSERVICE](#)

[TIBCControlAndQueryService](#)

[TIBCBackupRestoreService](#)

TIBCBackupService

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.3.1 Members

[TIBCBackupService](#) class overview.

Properties

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BackupFile	Used to set or return the backup file name.
BlockingFactor	Used to set the blocking factor for the tape device as an integer.
BufferSize (inherited from TIBCControlAndQueryService)	Used to set or return the buffer size.
Database	Used to set or return the database name.
Eof (inherited from TIBCControlAndQueryService)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.

NBackupLevel (inherited from TIBCBackupRestoreService)	Used to set backup level for nBackup.
NBackupOptions (inherited from TIBCBackupRestoreService)	Used to set backup options for nBackup.
Options	Used to specify the behaviour of a TIBCBackupService object.
Params (inherited from TCustomIBCService)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCService)	Used to select the network protocol.
Server (inherited from TCustomIBCService)	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB (inherited from TCustomIBCService)	Used to return and sets SPB parameters.
UseNBackup (inherited from TIBCBackupRestoreService)	Used to enable or disable using nBackup service.
Verbose	Used to set or return the backup in the verbose mode.

Methods

Name	Description
Attach (inherited from TCustomIBCService)	Attaches to the database.
Detach (inherited from TCustomIBCService)	Detaches from the database.
GetNextChunk (inherited from TIBCControlAndQueryService)	Returns the next chunk of data.
GetNextLine (inherited from TIBCControlAndQueryService)	Returns the next line of data.
ServiceStart (inherited from TCustomIBCService)	Starts the service.

Events

Name	Description
OnAttach (inherited from TCustomIBCSERVICE)	Occurs when the database is attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.3.2 Properties

Properties of the **TIBCBACKUPSERVICE** class.

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

Public

Name	Description
Eof (inherited from TIBCCONTROLANDQUERYSERVICE)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
NBackupLevel (inherited from TIBCBACKUPRESTORESERVICE)	Used to set backup level for nBackup.
NBackupOptions (inherited from TIBCBACKUPRESTORESERVICE)	Used to set backup options for nBackup.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.
UseNBackup (inherited from TIBCBACKUPRESTORESERVICE)	Used to enable or disable using nBackup service.

Published

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

Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BackupFile	Used to set or return the backup file name.
BlockingFactor	Used to set the blocking factor for the tape device as an integer.
BufferSize (inherited from TIBCCONTROLANDQUERYSERVICE)	Used to set or return the buffer size.
Database	Used to set or return the database name.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Options	Used to specify the behaviour of a TIBCBACKUPSERVICE object.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
Verbose	Used to set or return the backup in the verbose mode.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.3.2.1 BackupFile Property

Used to set or return the backup file name.

Class

[TIBCBackupService](#)

Syntax

```
property BackupFile: TStrings;
```

Remarks

Use the BackupFile property to set or return the backup file name.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.3.2.2 BlockingFactor Property

Used to set the blocking factor for the tape device as an integer.

Class

[TIBCBackupService](#)

Syntax

```
property BlockingFactor: Integer default 0;
```

Remarks

Use the BlockingFactor property to set the blocking factor for the tape device as an integer.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.3.2.3 Database Property

Used to set or return the database name.

Class

[TIBCBackupService](#)

Syntax

```
property Database: string;
```

Remarks

Use the Database property to set or return the database name to set properties on.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.3.2.4 Options Property

Used to specify the behaviour of a TIBCBackupService object.

Class

[TIBCBackupService](#)

Syntax

```
property options: TIBCBackupOptions default [];
```

Remarks

Set the properties of Options to specify the behaviour of a TIBCBackupService object.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.3.2.5 Verbose Property

Used to set or return the backup in the verbose mode.

Class

[TIBCBackupService](#)

Syntax

```
property verbose: Boolean;
```

Remarks

Use the Verbose property to set or return the backup in the verbose mode.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.4 TIBConfigParams Class

TIBConfigParams holds the configuration information about an InterBase server.

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

Unit

[IBAdmin](#)

Syntax

```
TIBConfigParams = class(System.TObject);
```

Remarks

TIBConfigParams holds the configuration information about an InterBase server. The TIBConfigParams type stores server configuration settings.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.4.1 Members

[TIBConfigParams](#) class overview.

Properties

Name	Description
BaseLocation	Used to determine the base location.
LockFileLocation	Used to determine the lock file location.
MessageFileLocation	Used to determine the message file location.
SecurityDatabaseLocation	Used to determine the security database location.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.4.2 Properties

Properties of the **TIBCConfigParams** class.

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

Public

Name	Description
BaseLocation	Used to determine the base location.
LockFileLocation	Used to determine the lock file location.
MessageFileLocation	Used to determine the message file location.
SecurityDatabaseLocation	Used to determine the security database location.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.4.2.1 BaseLocation Property

Used to determine the base location.

Class

[TIBCConfigParams](#)

Syntax

```
property BaseLocation: string;
```

Remarks

Use the BaseLocation property to determine the base location.

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.12.1.4.2.2 LockFileLocation Property

Used to determine the lock file location.

Class

[TIBConfigParams](#)

Syntax

```
property LockFileLocation: string;
```

Remarks

Use the LockFileLocation property to determine the lock file location.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.4.2.3 MessageFileLocation Property

Used to determine the message file location.

Class

[TIBConfigParams](#)

Syntax

```
property MessageFileLocation: string;
```

Remarks

Use the MessageFileLocation to determine the message file location.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.4.2.4 SecurityDatabaseLocation Property

Used to determine the security database location.

Class

[TIBConfigParams](#)

Syntax

```
property SecurityDatabaseLocation: string;
```

Remarks

Use the SecurityDatabaseLocation to determine the security database location.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5 TIBConfigService Class

Use a TIBConfigService object to configure database parameters.

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

Unit

[IBAdmin](#)

Syntax

```
TIBConfigService = class(TCustomIBService);
```

Remarks

Use a TIBConfigService component to configure database parameters, including page buffers, access mode, and sweep interval.

Note: This feature is available in InterBase only.

Inheritance Hierarchy

[TCustomIBService](#)

TIBConfigService

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.12.1.5.1 Members

[TIBConfigService](#) class overview.

Properties

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
Connection	Used to specify the connection in which the dataset will be executed.
Database	Used to set or return the database name.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
JournalInformation	Holds information about journal system.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.
Transaction	Used to determine the transaction under which the query of this dataset executes.

Methods

Name	Description
ActivateShadow	Activates the database

	shadow.
AlterJournal	Alters a pre-existing journal system.
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
BringDatabaseOnline	Brings a database online.
CreateJournal	Creates a journal.
CreateJournalArchive	Creates an archive.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
DisableFlush	Disables database flushing.
DropJournal	Drops a journal system.
DropJournalArchive	Drops an archive.
FlushDatabase	Flushes a database.
GetJournalInformation	Retrieves journaling information.
ReclaimMemory	Used to reclaim memory.
ServiceStart	Is not currently used by TIBConfigService.
SetAsyncMode	Changes the database write mode to asynchronous (buffered writes).
SetDBSqlDialect	Sets the SQL dialect for the database.
SetFlushInterval	Sets the interval for database flushing
SetGroupCommit	Sets group commit.
SetLingerInterval	Sets the linger interval.
SetPageBuffers	Sets the number of database page buffers.
SetReadOnly	Sets the database to read only.
SetReclaimInterval	Sets the reclaim interval.
SetReserveSpace	Reserves space for versioning.
SetSweepInterval	Sets the sweep interval for the database.

ShutdownDatabase	Shuts down the database.
SweepDatabase	Performs a database sweep.

Events

Name	Description
OnAttach (inherited from TCustomIBCSERVICE)	Occurs when the database is attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.2 Properties

Properties of the **TIBConfigService** class.

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

Public

Name	Description
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.

Published

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
Connection	Used to specify the connection in which the dataset will be executed.
Database	Used to set or return the database name.
JournalInformation	Holds information about journal system.

LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
Transaction	Used to determine the transaction under which the query of this dataset executes.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.2.1 Connection Property

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

Class

[TIBConfigService](#)

Syntax

```
property Connection: TIBConnection;
```

Remarks

Use the Connection property to specify the connection in which the service will be executed. If connection is not connected, the Open method calls Connection.Connect.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.12.1.5.2.2 Database Property

Used to set or return the database name.

Class

[TIBConfigService](#)

Syntax

```
property Database: string;
```

Remarks

Use the Database property to set or return the database name to set properties on.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.2.3 JournalInformation Property

Holds information about journal system.

Class

[TIBConfigService](#)

Syntax

```
property JournalInformation: TIBJournalInformation;
```

Remarks

The JournalInformation property holds information about a journal system. Gives access to the underlying IBCJournalInformation field.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.2.4 Transaction Property

Used to determine the transaction under which the query of this dataset executes.

Class

[TIBConfigService](#)

Syntax

```
property Transaction: TIBTransaction;
```

Remarks

Use the Transaction property to determine the transaction under which the query of this dataset executes. You can separately set transaction for executing modifying queries with the [TCustomIBCDataSet.UpdateTransaction](#) property. By default the Transaction and the UpdateTransaction properties are the same.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3 Methods

Methods of the **TIBConfigService** class.

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

Public

Name	Description
ActivateShadow	Activates the database shadow.
AlterJournal	Alters a pre-existing journal system.
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
BringDatabaseOnline	Brings a database online.
CreateJournal	Creates a journal.
CreateJournalArchive	Creates an archive.

Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
DisableFlush	Disables database flushing.
DropJournal	Drops a journal system.
DropJournalArchive	Drops an archive.
FlushDatabase	Flushes a database.
GetJournalInformation	Retrieves journaling information.
ReclaimMemory	Used to reclaim memory.
ServiceStart	Is not currently used by TIBCCONFIGSERVICE.
SetAsyncMode	Changes the database write mode to asynchronous (buffered writes).
SetDBSQLDialect	Sets the SQL dialect for the database.
SetFlushInterval	Sets the interval for database flushing
SetGroupCommit	Sets group commit.
SetLingerInterval	Sets the linger interval.
SetPageBuffers	Sets the number of database page buffers.
SetReadOnly	Sets the database to read only.
SetReclaimInterval	Sets the reclaim interval.
SetReserveSpace	Reserves space for versioning.
SetSweepInterval	Sets the sweep interval for the database.
ShutdownDatabase	Shuts down the database.
SweepDatabase	Performs a database sweep.

See Also

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

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.1 ActivateShadow Method

Activates the database shadow.

Class

[TIBConfigService](#)

Syntax

```
procedure ActivateShadow;
```

Remarks

Call the ActivateShadow method to activate the database shadow.

InterBase 6 lets you recover a database in case of disk failure, network failure, or accidental deletion of the database. This recovery method is known as disk shadowing, or simply shadowing. Before you can activate shadowing, you must first create a database shadow, as discussed in 'Shadowing' and 'Creating a shadow' in the InterBase 6 Operations Guide.

Note: You should install InterBase 6 to use this feature.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.2 AlterJournal Method

Alters a pre-existing journal system.

Class

[TIBConfigService](#)

Syntax

```
procedure AlterJournal;
```

Remarks

Call the AlterJournal method to alter a pre-existing journal system.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.3 BringDatabaseOnline Method

Brings a database online.

Class

[TIBConfigService](#)

Syntax

```
procedure BringDatabaseOnline(Options: TIBCSutdownOptions = soNormal);
```

Parameters

Options

Remarks

Call the BringDatabaseOnline method to bring a database online.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.4 CreateJournal Method

Creates a journal.

Class

[TIBConfigService](#)

Syntax

```
procedure CreateJournal;
```

Remarks

Call the CreateJournal method to create a journal based on JournalInformation.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.5 CreateJournalArchive Method

Creates an archive.

Class

[TIBConfigService](#)

Syntax

```
procedure CreateJournalArchive(const Directory: string);
```

Parameters

Directory

Holds the directory in which the archive is situated.

Remarks

Call the CreateJournalArchive method to create an archive.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.6 DisableFlush Method

Disables database flushing.

Class

[TIBConfigService](#)

Syntax

```
procedure DisableFlush;
```

Remarks

Call the DisableFlush method to disable database flushing.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.7 DropJournal Method

Drops a journal system.

Class

[TIBConfigService](#)

Syntax

```
procedure DropJournal;
```

Remarks

Call the DropJournal method to drop a journal system.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.8 DropJournalArchive Method

Drops an archive.

Class

[TIBConfigService](#)

Syntax

```
procedure DropJournalArchive;
```

Remarks

Call the DropJournalArchive to drop an archive.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.12.1.5.3.9 FlushDatabase Method

Flushes a database.

Class

[TIBConfigService](#)

Syntax

```
procedure FlushDatabase;
```

Remarks

Call the FlushDataBase method to flush a database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.10 GetJournalInformation Method

Retrieves journaling information.

Class

[TIBConfigService](#)

Syntax

```
procedure GetJournalInformation;
```

Remarks

Call the GetJournalInformation method to retrieve journaling information for this database and stores it in the [JournalInformation](#) property.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.11 ReclaimMemory Method

Used to reclaim memory.

Class

[TIBConfigService](#)

Syntax

```
procedure ReclaimMemory;
```

Remarks

Use the ReclaimMemory property to reclaim memory.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.12 ServiceStart Method

Is not currently used by TIBConfigService.

Class

[TIBConfigService](#)

Syntax

```
procedure ServiceStart; override;
```

Remarks

Do not use ServiceStart. TIBConfigService does not currently use this method.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.13 SetAsyncMode Method

Changes the database write mode to asynchronous (buffered writes).

Class

[TIBConfigService](#)

Syntax

```
procedure SetAsyncMode(Value: Boolean);
```

Parameters

Value

True, if the database write mode is changed to asynchronous.

Remarks

Set the SetAsyncMode method to True to change the database write mode to asynchronous (buffered writes).

InterBase 6 allows you to write to databases in both synchronous and asynchronous modes. In synchronous mode, the database writes are forced. In asynchronous mode, the database writes are buffered.

For more information, refer to 'Forced writes vs. buffered writes' in the InterBase 6 Operations Guide.

Note: You could install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.14 SetDBSqlDialect Method

Sets the SQL dialect for the database.

Class

[TIBConfigService](#)

Syntax

```
procedure SetDBSqlDialect(Value: Integer);
```

Parameters*Value*

Holds the appropriate values of SQL dialect (valid values are 1, 2, and 3).

Remarks

Call the SetDBSqlDialect method to set the SQL dialect for the database. Valid integer values are 1, 2, and 3.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.15 SetFlushInterval Method

Sets the interval for database flushing

Class

[TIBConfigService](#)

Syntax

```
procedure SetFlushInterval(value: Integer);
```

Parameters*Value*

Holds the interval.

Remarks

Call the SetFlushInterval method to set the interval for database flushing.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.16 SetGroupCommit Method

Sets group commit.

Class

[TIBConfigService](#)

Syntax

```
procedure SetGroupCommit(Value: Boolean);
```

Parameters

Value

Remarks

Call the SetGroupCommit method to set group commit.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.17 SetLingerInterval Method

Sets the linger interval.

Class

[TIBConfigService](#)

Syntax

```
procedure SetLingerInterval(Value: Integer);
```

Parameters

Value

Holds the linger interval.

Remarks

Call the SetLingerInterval method to set the linger interval.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.18 SetPageBuffers Method

Sets the number of database page buffers.

Class

[TIBConfigService](#)

Syntax

```
procedure SetPageBuffers(Value: Integer);
```

Parameters

Value

Holds the number of database page buffers.

Remarks

Call the SetPageBuffers method to set the number of database page buffers.

When a program establishes a connection to a database, InterBase allocates system memory to use as a private buffer. The buffers are used to store accessed database pages to speed performance. The number of buffers assigned determines how many simultaneous database pages it can have access to in the memory pool. Buffers remain assigned until a program finishes with a database.

For more information on page buffers, refer to 'Setting database cache buffers' in the InterBase 6 Programmer's Guide.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.19 SetReadOnly Method

Sets the database to read only.

Class

[TIBConfigService](#)

Syntax

```
procedure SetReadOnly(Value: Boolean);
```

Parameters

Value

True, if the database is set to read only.

Remarks

Call the SetReadOnly method to set the database to read only.

Access mode specifies the type of access a transaction has for the table it uses. There are two possible modes: read-only and read-write.

Read-only specifies that a transaction can read data from a table, but cannot insert, update, or delete table data. Read-write specifies that a transaction can select, insert, update, and delete table data. This is the default setting if none is specified.

Tip: You should specify the transaction's access mode even if it is read-write. It makes the application's source code easier to read and debug, because the program's intentions are clearly spelled out.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.20 SetReclaimInterval Method

Sets the reclaim interval.

Class

[TIBConfigService](#)

Syntax

```
procedure SetReclaimInterval(Value: Integer);
```

Parameters

Value

Holds the reclaim interval.

Remarks

Call the SetReclaimInterval method to set the reclaim interval.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.21 SetReserveSpace Method

Reserves space for versioning.

Class

[TIBConfigService](#)

Syntax

```
procedure SetReserveSpace(Value: Boolean);
```

Parameters

Value

If True, space for versioning is reserved.

Remarks

Set SetReserveSpace to true to reserve space on the data page for versioning.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.22 SetSweepInterval Method

Sets the sweep interval for the database.

Class

[TIBConfigService](#)

Syntax

```
procedure SetSweepInterval(Value: Integer);
```

Parameters

Value

Holds the sweep interval.

Remarks

Call `SetSweepInterval` set the sweep interval for the database. The sweep interval refers to the number of transactions between database sweeps.

To turn off database sweeps, set the sweep interval to 0.

Sweeping a database is a systematic way of removing outdated records from the database. Periodic sweeping keeps the database from getting too large. However, sweeping can also slow performance. For more information, refer to 'Setting the sweep interval' in the InterBase 6 Operations Guide.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.23 ShutdownDatabase Method

Shuts down the database.

Class

[TIBCConfigService](#)

Syntax

```
procedure ShutdownDatabase(Mode: TIBCShutdownMode; wait: Integer;  
Options: TIBCShutdownOptions = soNormal);
```

Parameters

Mode

Wait

Holds the number of seconds to wait before shutting the database.

Options

Remarks

Call the `ShutdownDatabase` method to shut down the database after a specified number of seconds.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.5.3.24 SweepDatabase Method

Performs a database sweep.

Class

[TIBConfigService](#)

Syntax

```
procedure SweepDatabase;
```

Remarks

Call the SweepDatabase method to perform a database sweep.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.6 TIBControlAndQueryService Class

TIBControlAndQueryService is the base class from which the log, statistical, validation, security, and backup and restore TIBService components descend.

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

Unit

[IBAdmin](#)

Syntax

```
TIBControlAndQueryService = class(TCustomIBService);
```

Remarks

TIBControlAndQueryService is the base class from which the log, statistical, validation,

security, and backup and restore TIBCSERVICE components descend. The service is unstructured because the output is in an unformatted text file.

Inheritance Hierarchy

[TCustomIBCSERVICE](#)

TIBCSERVICEControlAndQuerySERVICE

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.6.1 Members

[TIBCSERVICEControlAndQuerySERVICE](#) class overview.

Properties

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BufferSize	Used to set or return the buffer size.
Eof	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.

Methods

Name	Description
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
GetNextChunk	Returns the next chunk of data.
GetNextLine	Returns the next line of data.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.

Events

Name	Description
OnAttach (inherited from TCustomIBCSERVICE)	Occurs when the database is attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.6.2 Properties

Properties of the **TIBCControlAndQueryService** class.

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

Public

Name	Description
Eof	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.

Published

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

Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BufferSize	Used to set or return the buffer size.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.6.2.1 BufferSize Property

Used to set or return the buffer size.

Class

[TIBControlAndQueryService](#)

Syntax

```
property BufferSize: Integer;
```

Remarks

Use the BufferSize property to set or return the buffer size.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.6.2.2 Eof Property

Used to determine whether the end of the file has been reached.

Class

[TIBControlAndQueryService](#)

Syntax

```
property Eof: Boolean;
```

Remarks

Use the Eof property to determine whether the end of the file has been reached.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.6.3 Methods

Methods of the **TIBControlAndQueryService** class.

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

Public

Name	Description
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
GetNextChunk	Returns the next chunk of data.
GetNextLine	Returns the next line of data.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.

See Also

- [TIBControlAndQueryService Class](#)

- [TIBControlAndQueryService Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.6.3.1 GetNextChunk Method

Returns the next chunk of data.

Class

[TIBControlAndQueryService](#)

Syntax

```
function GetNextChunk: string;
```

Remarks

Call the GetNextChunk method to get the next chunk of data.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.6.3.2 GetNextLine Method

Returns the next line of data.

Class

[TIBControlAndQueryService](#)

Syntax

```
function GetNextLine: string;
```

Remarks

Call the GetNextLine method to get the next line of data.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.12.1.7 TIBCDatabaseInfo Class

Describes an InterBase database.

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

Unit

[IBAdmin](#)

Syntax

```
TIBCDatabaseInfo = class(System.TObject);
```

Remarks

TIBCDatabaseInfo describes an InterBase database.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.7.1 Members

[TIBCDatabaseInfo](#) class overview.

Properties

Name	Description
DbName	Holds the name of the database.
NoOfAttachments	Holds the number of attachments.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.7.2 Properties

Properties of the **TIBCDatabaseInfo** class.

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

Public

Name	Description
DbName	Holds the name of the database.
NoOfAttachments	Holds the number of attachments.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.7.2.1 DbName Property

Holds the name of the database.

Class

[TIBCDatabaseInfo](#)

Syntax

```
property DbName: TStringDynArray;
```

Remarks

The DbName property holds the name of the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.7.2.2 NoOfAttachments Property

Holds the number of attachments.

Class

[TIBCDatabaseInfo](#)

Syntax

```
property NoOfAttachments: integer;
```

Remarks

The NoOfAttachments property holds the number of attachments.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.8 TIBCJournalInformation Class

A class used to access the TIBCConfigService component properties.

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

Unit

[IBCAdmin](#)

Syntax

```
TIBCJournalInformation = class(TComponent);
```

Remarks

The TIBCJournalInformation class is used for displaying and setting Journal settings on a database. This class is used to access the TIBCConfigService component properties.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.8.1 Members

[TIBCJournalInformation](#) class overview.

Properties

Name	Description
CheckpointInterval	Used to determine the number of seconds between database checkpoints.

CheckpointLength	Used to determine the number of journal pages to be written before initiating a database checkpoint.
Directory	Holds the name of the directory in which journal is situated.
HasArchive	Turns archiving on.
HasJournal	Turns journaling on.
PageCache	Determines the number of journal buffers that will be allocated.
PageLength	Used to retrieve the page length.
PageSize	Determines the size of a journal page in bytes.
TimestampName	Determines whether or not to append the file creation timestamp to the base journal file name.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.8.2 Properties

Properties of the **TIBCJournalInformation** class.

For a complete list of the **TIBCJournalInformation** class members, see the

[TIBCJournalInformation Members](#) topic.

Published

Name	Description
CheckpointInterval	Used to determine the number of seconds between database checkpoints.
CheckpointLength	Used to determine the number of journal pages to be written before initiating a database checkpoint.
Directory	Holds the name of the

	directory in which journal is situated.
HasArchive	Turns archiving on.
HasJournal	Turns journaling on.
PageCache	Determines the number of journal buffers that will be allocated.
PageLength	Used to retrieve the page length.
PageSize	Determines the size of a journal page in bytes.
TimestampName	Determines whether or not to append the file creation timestamp to the base journal file name.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.8.2.1 CheckpointInterval Property

Used to determine the number of seconds between database checkpoints.

Class

[TIBCJournalInformation](#)

Syntax

```
property CheckpointInterval: Integer default 0;
```

Remarks

Use the CheckpointInterval property to determine the number of seconds between database checkpoints.

Note: If both CHECKPOINT LENGTH and CHECKPOINT INTERVAL are specified, whichever

event occurs first will initiate a database checkpoint.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.8.2.2 CheckpointLength Property

Used to determine the number of journal pages to be written before initiating a database checkpoint.

Class

[TIBCJournalInformation](#)

Syntax

```
property CheckpointLength: Integer default 500;
```

Remarks

Use the CheckpointInterval property to determine the number of journal pages to be written before initiating a database checkpoint.

Note: If both CHECKPOINT LENGTH and CHECKPOINT INTERVAL are specified, whichever event occurs first will initiate a database checkpoint.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.8.2.3 Directory Property

Holds the name of the directory in which journal is situated.

Class

[TIBCJournalInformation](#)

Syntax

```
property Directory: string;
```

Remarks

The Directory property holds the name of the directory in which journal is situated.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.8.2.4 HasArchive Property

Turns archiving on.

Class

[TIBCJournalInformation](#)

Syntax

```
property HasArchive: Boolean;
```

Remarks

Set the HasArchive method to True to turn archiving on.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.8.2.5 HasJournal Property

Turns journaling on.

Class

[TIBCJournalInformation](#)

Syntax

```
property HasJournal: Boolean;
```

Remarks

Set the HasJournal method to True to turn journaling on.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.8.2.6 PageCache Property

Determines the number of journal buffers that will be allocated.

Class

[TIBCJournalInformation](#)

Syntax

```
property PageCache: Integer default 100;
```

Remarks

The PageCache property determines the number of journal buffers that will be allocated. The size of each buffer is the same as the journal page size.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.8.2.7 PageLength Property

Used to retrieve the page length.

Class

[TIBCJournalInformation](#)

Syntax

```
property PageLength: Integer default 500;
```

Remarks

Use the PageLength property to retrieve the page length.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.8.2.8 PageSize Property

Determines the size of a journal page in bytes.

Class

[TIBCJournalInformation](#)

Syntax

```
property PageSize: Integer default 0;
```

Remarks

The PageSize property determines the size of a journal page in bytes. A journal page size must be at least twice the size of a database page size. If a journal page size of less is specified, it will be rounded up to twice the database page size and a warning will be returned.

The journal page size need not be a power of 2.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.8.2.9 TimestampName Property

Determines whether or not to append the file creation timestamp to the base journal file name.

Class

[TIBCJournalInformation](#)

Syntax

```
property TimestampName: Boolean default True;
```

Remarks

The TimestampName property determines whether or not to append the file creation timestamp to the base journal file name.

If this option is on, the base journal file name will be appended with a timestamp of the form:

<YYYY>_<MM>_<DD>T<hh>_<mm>_<ss>Z.<sequence-number>.journal

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.9 TIBCLicenseInfo Class

Stores information about licensed users.

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

Unit

[IBCAdmin](#)

Syntax

```
TIBCLicenseInfo = class(System.TObject);
```

Remarks

TIBCLicenseInfo stores information about licensed users.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.9.1 Members

[TIBCLicenseInfo](#) class overview.

Properties

Name	Description
Desc	Used to list the description of each licensed user.
Id	Used to list the user Id for each licensed user.
Key	Used to list the license key for each licensed user.
LicensedUsers	Used to determine the number of users.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.9.2 Properties

Properties of the **TIBCLicenseInfo** class.

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

Public

Name	Description
Desc	Used to list the description of each licensed user.
Id	Used to list the user Id for each licensed user.
Key	Used to list the license key for each licensed user.
LicensedUsers	Used to determine the number of users.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.9.2.1 Desc Property

Used to list the description of each licensed user.

Class

[TIBCLicenseInfo](#)

Syntax

```
property Desc: TStringDynArray;
```

Remarks

The Desc property lists the description of each licensed user.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.9.2.2 Id Property

Used to list the user Id for each licensed user.

Class

[TIBCLicenseInfo](#)

Syntax

```
property Id: TStringDynArray;
```

Remarks

The Id property lists the user Id for each licensed user.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.9.2.3 Key Property

Used to list the license key for each licensed user.

Class

[TIBCLicenseInfo](#)

Syntax

```
property Key: TStringDynArray;
```

Remarks

The Key property lists the license key for each licensed user.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.9.2.4 LicensedUsers Property

Used to determine the number of users.

Class

[TIBCLicenseInfo](#)

Syntax

```
property LicensedUsers: integer;
```

Remarks

The LicensedUsers property indicates the number of users (the number of entries in the lists).

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.10 TIBCLicenseMaskInfo Class

Indicates the software activation certificate options enabled on the server.

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

Unit

[IBCAAdmin](#)

Syntax

```
TIBCLicenseMaskInfo = class(System.TObject);
```

Remarks

TIBCLicenseMaskInfo indicates the software activation certificate options enabled on the server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.10.1 Members

[TIBCLicenseMaskInfo](#) class overview.

Properties

Name	Description
CapabilityMask	Used to determine a bitmask representing the

	capabilities currently enabled on the server.
LicenseMask	Used to determine a bitmask representing the software activation certificate options currently enabled on the server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.10.2 Properties

Properties of the **TIBCLicenseMaskInfo** class.

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

Public

Name	Description
CapabilityMask	Used to determine a bitmask representing the capabilities currently enabled on the server.
LicenseMask	Used to determine a bitmask representing the software activation certificate options currently enabled on the server.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.10.2.1 CapabilityMask Property

Used to determine a bitmask representing the capabilities currently enabled on the server.

Class

[TIBCLicenseMaskInfo](#)

Syntax

```
property CapabilityMask: integer;
```

Remarks

Use the CapabilityMask to determine a bitmask representing the capabilities currently enabled on the server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.10.2.2 LicenseMask Property

Used to determine a bitmask representing the software activation certificate options currently enabled on the server.

Class

[TIBCLicenseMaskInfo](#)

Syntax

```
property LicenseMask: integer;
```

Remarks

Use the LicenseMask property to determine a bitmask representing the software activation certificate options currently enabled on the server.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.11 TIBCLicensingService Class

TIBCLicensingService configures the licensing parameters

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

Unit

[IBCAAdmin](#)

Syntax

```
TIBCLicensingService = class(TCustomIBCSERVICE);
```

Remarks

Use the TIBCLicensingService component to add or remove InterBase software activation certificates.

Inheritance Hierarchy

[TCustomIBCSERVICE](#)

TIBCLicensingService

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.11.1 Members

[TIBCLicensingService](#) class overview.

Properties

Name	Description
Action	Used to indicate what action the licensing service should take.
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
ID	Used to set or return the license identification.

Key	Used to set or return the license key.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.

Methods

Name	Description
AddLicense	Adds an InterBase software activation certificate.
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
RemoveLicense	Removes an InterBase software activation certificate.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.

Events

Name	Description
OnAttach (inherited from TCustomIBCSERVICE)	Occurs when the database is attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.11.2 Properties

Properties of the **TIBCLicensingService** class.

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

Public

Name	Description
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.

Published

Name	Description
Action	Used to indicate what action the licensing service should take.
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
ID	Used to set or return the license identification.
Key	Used to set or return the license key.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.

See Also

- [TIBCLicensingService Class](#)

- [TIBCLicensingService Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.11.2.1 Action Property

Used to indicate what action the licensing service should take.

Class

[TIBCLicensingService](#)

Syntax

```
property Action: TIBCLicensingAction default 1aAdd;
```

Remarks

Use the Action property to indicate what action the licensing service should take. This causes the service to call the [AddLicense](#) or [RemoveLicense](#) method, depending on whether the software activation certificate is added or removed.

Note: You should install InterBase 6 to use this feature.

See Also

- [AddLicense](#)
- [RemoveLicense](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.11.2.2 ID Property

Used to set or return the license identification.

Class

[TIBCLicensingService](#)

Syntax

```
property ID: string;
```

Remarks

Use the ID property to set or return the license identification.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.11.2.3 Key Property

Used to set or return the license key.

Class

[TIBCLicensingService](#)

Syntax

```
property Key: string;
```

Remarks

Use the Key property to set or return the license key.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.11.3 Methods

Methods of the **TIBCLicensingService** class.

For a complete list of the **TIBCLicensingService** class members, see the

[TIBCLicensingService Members](#) topic.

Public

Name	Description
AddLicense	Adds an InterBase software activation certificate.
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.

Detach (inherited from TCustomIBService)	Detaches from the database.
RemoveLicense	Removes an InterBase software activation certificate.
ServiceStart (inherited from TCustomIBService)	Starts the service.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.11.3.1 AddLicense Method

Adds an InterBase software activation certificate.

Class

[TIBCLicensingService](#)

Syntax

```
procedure AddLicense;
```

Remarks

Call the AddLicense method along to add an InterBase software activation certificate. The [Key](#) and [ID](#) properties should be supplied.

Note: You could install InterBase 6 to use this feature.

See Also

- [Key](#)
- [ID](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.11.3.2 RemoveLicense Method

Removes an InterBase software activation certificate.

Class

[TIBCLicensingService](#)

Syntax

```
procedure RemoveLicense;
```

Remarks

Call the RemoveLicense method to remove an InterBase software activation certificate.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.12 TIBCLimboTransactionInfo Class

TIBCLimboTransactionInfo stores information about a limbo transaction.

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

Unit

[IBAdmin](#)

Syntax

```
TIBCLimboTransactionInfo = class(System.TObject);
```

Remarks

TIBCLimboTransactionInfo describes limbo transaction. Limbo transactions are usually caused by the failure of a two-phased commit. They can also exist due to system failure or when a single-database transaction is prepared.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.12.1 Members

[TIBCLimboTransactionInfo](#) class overview.

Properties

Name	Description
Action	Used to retrieve the action to take for ending the transaction.
Advise	Used to retrieve the suggested action for ending the transaction.
HostSite	Holds the host name for the server where the transaction was started.
ID	Holds the identifier for the limbo transaction.
MultiDatabase	Used to indicate whether the limbo transaction involves multiple databases.
RemoteDatabasePath	Holds the path to the remote server.
RemoteSite	Holds the host name for a remote server involved in the transaction.
State	Used to indicate the current state of the limbo transaction.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.12.2 Properties

Properties of the **TIBCLimboTransactionInfo** class.

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

Public

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

Action	Used to retrieve the action to take for ending the transaction.
Advise	Used to retrieve the suggested action for ending the transaction.
HostSite	Holds the host name for the server where the transaction was started.
ID	Holds the identifier for the limbo transaction.
MultiDatabase	Used to indicate whether the limbo transaction involves multiple databases.
RemoteDatabasePath	Holds the path to the remote server.
RemoteSite	Holds the host name for a remote server involved in the transaction.
State	Used to indicate the current state of the limbo transaction.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.12.2.1 Action Property

Used to retrieve the action to take for ending the transaction.

Class

[TIBCLimboTransactionInfo](#)

Syntax

```
property Action: TIBCTransactionAction;
```

Remarks

Use the Action property to show the action to take for ending the transaction.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.12.2.2 Advise Property

Used to retrieve the suggested action for ending the transaction.

Class

[TIBCLimboTransactionInfo](#)

Syntax

```
property Advise: TIBCTransactionAdvise;
```

Remarks

Use the Advise property to show the suggested action for ending the transaction.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.12.2.3 HostSite Property

Holds the host name for the server where the transaction was started.

Class

[TIBCLimboTransactionInfo](#)

Syntax

```
property HostSite: string;
```

Remarks

The HostSite property holds the host name for the server where the transaction was started.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.12.2.4 ID Property

Holds the identifier for the limbo transaction.

Class

[TIBCLimboTransactionInfo](#)

Syntax

```
property ID: integer;
```

Remarks

The ID property holds the identifier for the limbo transaction.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.12.2.5 MultiDatabase Property

Used to indicate whether the limbo transaction involves multiple databases.

Class

[TIBCLimboTransactionInfo](#)

Syntax

```
property MultiDatabase: boolean;
```

Remarks

Use the MultiDatabase property to indicate whether the limbo transaction involves multiple databases.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.12.2.6 RemoteDatabasePath Property

Holds the path to the remote server.

Class

[TIBCLimboTransactionInfo](#)

Syntax

```
property RemoteDatabasePath: string;
```

Remarks

The RemoteDatabasePath property holds the path to the remote server.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.12.2.7 RemoteSite Property

Holds the host name for a remote server involved in the transaction.

Class

[TIBCLimboTransactionInfo](#)

Syntax

```
property RemoteSite: string;
```

Remarks

The RemoteSite property holds the host name for a remote server involved in the transaction.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.12.2.8 State Property

Used to indicate the current state of the limbo transaction.

Class

[TIBCLimboTransactionInfo](#)

Syntax

```
property State: TIBCTransactionState;
```

Remarks

Use the State property to indicate the current state of the limbo transaction.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.13 TIBCLogService Class

Returns the contents of the interbase.log file from server.

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

Unit

[IBCAdmin](#)

Syntax

```
TIBCLogService = class(TIBControlAndQueryService);
```

Remarks

Use a TIBCLogService object to return the contents of the interbase.log file from server.

Inheritance Hierarchy

[TCustomIBCSERVICE](#)

[TIBControlAndQueryService](#)

TIBCLogService

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.13.1 Members

[TIBCLogService](#) class overview.

Properties

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BufferSize (inherited from	Used to set or return the buffer size.

TIBControlAndQueryService)	
Eof (inherited from TIBControlAndQueryService)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.

Methods

Name	Description
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
GetNextChunk (inherited from TIBControlAndQueryService)	Returns the next chunk of data.
GetNextLine (inherited from TIBControlAndQueryService)	Returns the next line of data.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.

Events

Name	Description
OnAttach (inherited from TCustomIBCSERVICE)	Occurs when the database is attached.

Devart. All Rights Reserved.

5.12.1.14 TIBCRestoreService Class

Used to restore a database.

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

Unit

[IBCAAdmin](#)

Syntax

```
TIBCRestoreService = class(TIBCBackupRestoreService);
```

Remarks

Use a TIBCRestoreService object to restore a database.

Inheritance Hierarchy

[TCustomIBCService](#)

[TIBCControlAndQueryService](#)

[TIBCBackupRestoreService](#)

TIBCRestoreService

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.14.1 Members

[TIBCRestoreService](#) class overview.

Properties

Name	Description
Active (inherited from TCustomIBCService)	Used to set the service to active or inactive (default).
BackupFile	Used to set or return the backup file name.
BufferSize (inherited from	Used to set or return the buffer size.

TIBControlAndQueryService)	
Database	Used to set or return the database name.
Eof (inherited from TIBControlAndQueryService)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
NBackupLevel (inherited from TIBBackupRestoreService)	Used to set backup level for nBackup.
NBackupOptions (inherited from TIBBackupRestoreService)	Used to set backup options for nBackup.
Options	Used to build restore options.
PageBuffers	Used to set or return the page buffer size.
PageSize	Used to set or return the page size.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.
UseNBackup (inherited from TIBBackupRestoreService)	Used to enable or disable using nBackup service.
Verbose	Used to set or return the restore in the verbose mode.

Methods

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

Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
GetNextChunk (inherited from TIBCControlAndQueryService)	Returns the next chunk of data.
GetNextLine (inherited from TIBCControlAndQueryService)	Returns the next line of data.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.

Events

Name	Description
OnAttach (inherited from TCustomIBCSERVICE)	Occurs when the database is attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.14.2 Properties

Properties of the **TIBCRestoreService** class.

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

Public

Name	Description
Eof (inherited from TIBCControlAndQueryService)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
NBackupLevel (inherited from TIBCBackupRestoreService)	Used to set backup level for nBackup.
NBackupOptions (inherited from TIBCBackupRestoreService)	Used to set backup options for nBackup.

TIBCBackupRestoreService)	
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.
UseNBackup (inherited from TIBCBackupRestoreService)	Used to enable or disable using nBackup service.

Published

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BackupFile	Used to set or return the backup file name.
BufferSize (inherited from TIBCControlAndQueryService)	Used to set or return the buffer size.
Database	Used to set or return the database name.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Options	Used to build restore options.
PageBuffers	Used to set or return the page buffer size.
PageSize	Used to set or return the page size.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
Verbose	Used to set or return the restore in the verbose mode.

See Also

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

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.14.2.1 BackupFile Property

Used to set or return the backup file name.

Class

[TIBCRestoreService](#)

Syntax

```
property BackupFile: TStrings;
```

Remarks

Use the BackupFile property to set or return the backup file name.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.14.2.2 Database Property

Used to set or return the database name.

Class

[TIBCRestoreService](#)

Syntax

```
property Database: TStrings;
```

Remarks

Use the Database property to set or return the database name to set properties on.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.12.1.14.2.3 Options Property

Used to build restore options.

Class

[TIBCRestoreService](#)

Syntax

```
property options: TIBCRestoreOptions default [roCreateNewDB];
```

Remarks

Use the Options property to build restore options of type TIBCRestoreOption into application.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.14.2.4 PageBuffers Property

Used to set or return the page buffer size.

Class

[TIBCRestoreService](#)

Syntax

```
property PageBuffers: Integer default 0;
```

Remarks

Use the PageBuffers property to set or return the page buffer size in kilobytes.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.14.2.5 PageSize Property

Used to set or return the page size.

Class

[TIBCRestoreService](#)

Syntax

```
property PageSize: Integer default 4096;
```

Remarks

Use the PageSize property to set or return the page size in kilobytes.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.14.2.6 Verbose Property

Used to set or return the restore in the verbose mode.

Class

[TIBCRestoreService](#)

Syntax

```
property Verbose: Boolean;
```

Remarks

Use the Verbose property to set or return the restore in the verbose mode.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15 TIBCSecurityService Class

Used to manage user access to the InterBase server.

For a list of all members of this type, see [TIBCSecurityService](#) members.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCSecurityService = class(TIBControlAndQueryService);
```

Remarks

Use a TIBCSecurityService object to manage user access to the InterBase server.

Inheritance Hierarchy

[TCustomIBCSERVICE](#)

[TIBControlAndQueryService](#)

TIBCSecurityService

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.1 Members

[TIBCSecurityService](#) class overview.

Properties

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BufferSize (inherited from TIBControlAndQueryService)	Used to set or return the buffer size.
Eof (inherited from TIBControlAndQueryService)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.

Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
SecurityAction	Used to determine the type of the operation that will be performed by InterBase Security Service.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.
UserDatabase	Used to add, modify, or delete a user in the specified database.
UserInfo	Used to return the user's information.
UserInfos	Used to get the user's information.
UserInfosCount	Used to get the number of returned records.

Methods

Name	Description
AddUser	Adds a user to the user table.
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
DeleteUser	Used to delete a user from the user table.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
DisplayUser	Fetches user information and dumps it into the TIBCUserInfo record
DisplayUsers	Fetches all valid user information.
EnableEUA	Enables user authentication.
GetNextChunk (inherited from TIBControlAndQueryService)	Returns the next chunk of data.

GetNextLine (inherited from TIBCControlAndQueryService)	Returns the next line of data.
ModifyUser	Modifies user's information.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.
SuspendEUA	Suspends embedded user authentication.

Events

Name	Description
OnAttach (inherited from TCustomIBCSERVICE)	Occurs when the database is attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.2 Properties

Properties of the **TIBCSecurityService** class.

For a complete list of the **TIBCSecurityService** class members, see the

[TIBCSecurityService Members](#) topic.

Public

Name	Description
Eof (inherited from TIBCControlAndQueryService)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.
UserInfos	Used to get the user's information.
UserInfosCount	Used to get the number of returned records.

Published

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BufferSize (inherited from TIBControlAndQueryService)	Used to set or return the buffer size.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
SecurityAction	Used to determine the type of the operation that will be performed by InterBase Security Service.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
UserDatabase	Used to add, modify, or delete a user in the specified database.
UserInfo	Used to return the user's information.

See Also

- [TIBSecurityService Class](#)
- [TIBSecurityService Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.2.1 SecurityAction Property

Used to determine the type of the operation that will be performed by InterBase Security Service.

Class

[TIBCSecurityService](#)

Syntax

```
property SecurityAction: TIBCSecurityAction default saAddUser;
```

Remarks

The SecurityAction property determines the type of the operation that will be performed by InterBase Security Service.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.2.2 UserDatabase Property

Used to add, modify, or delete a user in the specified database.

Class

[TIBCSecurityService](#)

Syntax

```
property UserDatabase: string;
```

Remarks

Set the UserDatabase property to add, modify, or delete a user in the specified database.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.2.3 UserInfo Property

Used to return the user's information.

Class

[TIBCSecurityService](#)

Syntax

```
property UserInfo: TIBCUserInfo;
```

Remarks

Fill in the UserInfo property to add, delete, or modify a user.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.2.4 UserInfos Property(Indexer)

Used to get the user's information.

Class

[TIBCSecurityService](#)

Syntax

```
property UserInfos[Index: Integer]: TIBCUserInfo;
```

Parameters

Index

Holds the index of the user to get information on.

Remarks

Use the UserInfos property to return the user's information as a TIBCUserInfo record, which contains the user name, groupID, UserID, and the user's first, middle, and last names.

Specify the index of the required user in the Index parameter.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.2.5 UserInfosCount Property

Used to get the number of returned records.

Class

[TIBCSecurityService](#)

Syntax

```
property UserInfosCount: Integer;
```

Remarks

Use the UserInfosCount to get the number of returned TIBCUserInfo records.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.3 Methods

Methods of the **TIBCSecurityService** class.

For a complete list of the **TIBCSecurityService** class members, see the [TIBCSecurityService Members](#) topic.

Public

Name	Description
AddUser	Adds a user to the user table.
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
DeleteUser	Used to delete a user from the user table.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
DisplayUser	Fetches user information and dumps it into the TIBCUserInfo record
DisplayUsers	Fetches all valid user information.
EnableEUA	Enables user authentication.
GetNextChunk (inherited from TIBCControlAndQueryService)	Returns the next chunk of data.
GetNextLine (inherited from	Returns the next line of data.

TIBCControlAndQueryService)	
ModifyUser	Modifies user's information.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.
SuspendEUA	Suspends embedded user authentication.

See Also

- [TIBCSecurityService Class](#)
- [TIBCSecurityService Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.3.1 AddUser Method

Adds a user to the user table.

Class

[TIBCSecurityService](#)

Syntax

```
procedure AddUser;
```

Remarks

Call the AddUser method to add a user to the user table.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.3.2 DeleteUser Method

Used to delete a user from the user table.

Class

[TIBCSecurityService](#)

Syntax

```
procedure DeleteUser;
```

Remarks

Call the DeleteUser method to delete a user from the user table.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.3.3 DisplayUser Method

Fetches user information and dumps it into the TIBCUserInfo record

Class

[TIBCSecurityService](#)

Syntax

```
procedure DisplayUser(const UserName: string);
```

Parameters

UserName

Holds the user name.

Remarks

Call the DisplayUser method to fetch user information and dump it into the TIBCUserInfo record, which can then be returned by the [UserInfos](#) property.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.3.4 DisplayUsers Method

Fetches all valid user information.

Class

[TIBCSecurityService](#)

Syntax

```
procedure DisplayUsers;
```

Remarks

Call the DisplayUsers method to fetch all valid user information, which can then be returned by the [UserInfo](#) property.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.3.5 EnableEUA Method

Enables user authentication.

Class

[TIBCSecurityService](#)

Syntax

```
procedure EnableEUA(Value: Boolean);
```

Parameters

Value

True if EUA is enabled.

Remarks

Call the EnableEUA method to enable embedded user authentication.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.3.6 ModifyUser Method

Modifies user's information.

Class

[TIBCSecurityService](#)

Syntax

```
procedure ModifyUser;
```

Remarks

Call ModifyUser to modify user's information.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.15.3.7 SuspendEUA Method

Suspends embedded user authentication.

Class

[TIBCSecurityService](#)

Syntax

```
procedure SuspendEUA(Value: Boolean);
```

Parameters

Value

True, if embedded user authentication is suspended.

Remarks

Call the SuspendEUA method to suspend embedded user authentication.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16 TIBCServerProperties Class

A class that returns database server information.

For a list of all members of this type, see [TIBCServerProperties](#) members.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCServerProperties = class(TCustomIBCSERVICE);
```

Remarks

The TIBCServerProperties class returns database server information, including configuration parameters, and also version and license information.

Note: You must install InterBase 6 to use this feature.

Inheritance Hierarchy

[TCustomIBCSERVICE](#)

TIBCServerProperties

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.1 Members

[TIBCServerProperties](#) class overview.

Properties

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
AliasCount	Used to get the alias count.
ConfigParams	Used to return server configuration parameters.
DatabaseInfo	Used to get database information.

Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
LicenseInfo	Used to get licensing information.
LicenseMaskInfo	Used to return licensing information as a TIBCLicenseMaskInfo record, which includes the license mask and capability mask.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.
VersionInfo	Used to get the server configuration parameters.

Methods

Name	Description
AddAlias	Adds database alias.
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
DeleteAlias	Deletes database alias.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
Fetch	Gets information from the server.
FetchAliasInfo	Returns database alias information.
FetchConfigParams	Returns database configuration parameters.
FetchDatabaseInfo	Returns database

	information.
FetchLicenseInfo	Returns database license information.
FetchLicenseMaskInfo	Returns license mask information.
FetchVersionInfo	Returns the database version information.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.

Events

Name	Description
OnAttach (inherited from TCustomIBCSERVICE)	Occurs when the database is attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.2 Properties

Properties of the **TIBCServerProperties** class.

For a complete list of the **TIBCServerProperties** class members, see the [TIBCServerProperties Members](#) topic.

Public

Name	Description
AliasCount	Used to get the alias count.
ConfigParams	Used to return server configuration parameters.
DatabaseInfo	Used to get database information.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
LicenseInfo	Used to get licensing information.
LicenseMaskInfo	Used to return licensing information as a TIBCLicenseMaskInfo record, which includes the

	license mask and capability mask.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.
VersionInfo	Used to get the server configuration parameters.

Published

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.

See Also

- [TIBCServerProperties Class](#)
- [TIBCServerProperties Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.2.1 AliasCount Property

Used to get the alias count.

Class

[TIBCServerProperties](#)

Syntax

```
property AliasCount: Integer;
```

Remarks

Use the AliasCount property to obtain the alias count.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.2.2 ConfigParams Property

Used to return server configuration parameters.

Class

[TIBCServerProperties](#)

Syntax

```
property ConfigParams: TIBConfigParams;
```

Remarks

Use the ConfigParams property to return the server configuration parameters as a TIBConfigParams record, which includes the configuration file data, base location, lock file location, message file location, and the security database location.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.2.3 DatabaseInfo Property

Used to get database information.

Class

[TIBCServerProperties](#)

Syntax

```
property DatabaseInfo: TIBCDatabaseInfo;
```

Remarks

Use the DatabaseInfo property to obtain database information, which includes the database name, number of attachments, and number of databases.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.2.4 LicenseInfo Property

Used to get licensing information.

Class

[TIBCServerProperties](#)

Syntax

```
property LicenseInfo: TIBCLicenseInfo;
```

Remarks

Use the LicenseInfo property to return licensing information as a TIBCLicenseInfo record, which includes the key, ID, and number of licensed users.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.2.5 LicenseMaskInfo Property

Used to return licensing information as a TIBCLicenseMaskInfo record, which includes the license mask and capability mask.

Class

[TIBCServerProperties](#)

Syntax

```
property LicenseMaskInfo: TIBCLicenseMaskInfo;
```

Remarks

Use the LicenseMaskInfo property to return licensing information as a TIBCLicenseMaskInfo record, which includes the license mask and capability mask. A license mask is a bitmask representing the software activation certificate options currently enabled on the server. A capability mask is a bitmask representing the capabilities currently enabled on the server.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.2.6 VersionInfo Property

Used to get the server configuration parameters.

Class

[TIBCServerProperties](#)

Syntax

```
property VersionInfo: TIBCVersionInfo;
```

Remarks

Use the VersionInfo property to obtain the server configuration parameters as a TIBCVersionInfo type, which includes the service version, and server version and implementation.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.3 Methods

Methods of the **TIBCServerProperties** class.

For a complete list of the **TIBCServerProperties** class members, see the

[TIBCServerProperties Members](#) topic.

Public

Name	Description
AddAlias	Adds database alias.
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
DeleteAlias	Deletes database alias.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
Fetch	Gets information from the server.
FetchAliasInfo	Returns database alias information.
FetchConfigParams	Returns database configuration parameters.
FetchDatabaseInfo	Returns database information.
FetchLicenseInfo	Returns database license information.
FetchLicenseMaskInfo	Returns license mask information.
FetchVersionInfo	Returns the database version information.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.

See Also

- [TIBCServerProperties Class](#)
- [TIBCServerProperties Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.12.1.16.3.1 AddAlias Method

Adds database alias.

Class

[TIBCServerProperties](#)

Syntax

```
procedure AddAlias(const Alias: string; const DBPath: string);
```

Parameters

Alias

Holds the database alias.

DBPath

Remarks

Call the AddAlias method to add database alias.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.3.2 DeleteAlias Method

Deletes database alias.

Class

[TIBCServerProperties](#)

Syntax

```
procedure DeleteAlias(const Alias: string);
```

Parameters

Alias

Holds database alias.

Remarks

Call the DeleteAlias method to delete database alias.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.3.3 Fetch Method

Gets information from the server.

Class

[TIBCServerProperties](#)

Syntax

```
procedure Fetch;
```

Remarks

Call the Fetch method to get information from the server.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.3.4 FetchAliasInfo Method

Returns database alias information.

Class

[TIBCServerProperties](#)

Syntax

```
procedure FetchAliasInfo;
```

Remarks

Call the FetchAliasInfo method to return database alias information.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.3.5 FetchConfigParams Method

Returns database configuration parameters.

Class

[TIBCServerProperties](#)

Syntax

```
procedure FetchConfigParams;
```

Remarks

Call the FetchConfigParams method to return database configuration parameters.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.3.6 FetchDatabaseInfo Method

Returns database information.

Class

[TIBCServerProperties](#)

Syntax

```
procedure FetchDatabaseInfo;
```

Remarks

Call the FetchDatabaseInfo method to return database information.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.3.7 FetchLicenseInfo Method

Returns database license information.

Class

[TIBCServerProperties](#)

Syntax

```
procedure FetchLicenseInfo;
```

Remarks

Call the FetchLicenseInfo method to return database license information.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.3.8 FetchLicenseMaskInfo Method

Returns license mask information.

Class

[TIBCServerProperties](#)

Syntax

```
procedure FetchLicenseMaskInfo;
```

Remarks

Call the FetchLicenseMaskInfo method to return database license mask information.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.16.3.9 FetchVersionInfo Method

Returns the database version information.

Class

[TIBCServerProperties](#)

Syntax

```
procedure FetchVersionInfo;
```

Remarks

Call the FetchVersionInfo method to return the database version information.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.17 TIBCStatisticalService Class

TIBCStatisticalService is used to view database statistics.

For a list of all members of this type, see [TIBCStatisticalService](#) members.

Unit

[IBAdmin](#)

Syntax

```
TIBCStatisticalService = class(TIBCControlAndQueryService);
```

Remarks

Use a TIBCStatisticalService object to view database statistics.

TIBCStatisticalService contains many properties and methods to allow you to build a statistical component into your application. Only the SYSDBA user or owner of the database will be able to run this service.

Inheritance Hierarchy

[TCustomIBCSERVICE](#)

[TIBCControlAndQueryService](#)

TIBCStatisticalService

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.17.1 Members

[TIBCStatisticalService](#) class overview.

Properties

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BufferSize (inherited from TIBCControlAndQueryService)	Used to set or return the buffer size.
Database	Used to set or return the database name.
Eof (inherited from TIBCControlAndQueryService)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Options	Used to specify the behaviour of a TIBCStatisticalService object.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.

TableNames	Used to specify the name of the table to request statistics for.
----------------------------	--

Methods

Name	Description
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
GetNextChunk (inherited from TIBCCControlAndQueryService)	Returns the next chunk of data.
GetNextLine (inherited from TIBCCControlAndQueryService)	Returns the next line of data.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.

Events

Name	Description
OnAttach (inherited from TCustomIBCSERVICE)	Occurs when the database is attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.17.2 Properties

Properties of the **TIBCStatisticalService** class.

For a complete list of the **TIBCStatisticalService** class members, see the

[TIBCStatisticalService Members](#) topic.

Public

Name	Description
Eof (inherited from TIBCCControlAndQueryService)	Used to determine whether the end of the file has been reached.

Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.

Published

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BufferSize (inherited from TIBCCControlAndQueryService)	Used to set or return the buffer size.
Database	Used to set or return the database name.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Options	Used to specify the behaviour of a TIBCStatisticalService object.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
TableNames	Used to specify the name of the table to request statistics for.

See Also

- [TIBCStatisticalService Class](#)
- [TIBCStatisticalService Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.17.2.1 Database Property

Used to set or return the database name.

Class

[TIBCStatisticalService](#)

Syntax

```
property Database: string;
```

Remarks

Use the Database property to set or return the database name to set properties on.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.17.2.2 Options Property

Used to specify the behaviour of a TIBCStatisticalService object.

Class

[TIBCStatisticalService](#)

Syntax

```
property Options: TIBCStatOptions;
```

Remarks

Use the Options property of TIBCStatisticalService to request database statistics.

These options are incremental; that is, setting DbLog to True also returns HeaderPages statistics, setting IndexPages to True returns also returns DbLog and HeaderPages statistics, and so forth.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.17.2.3 TableNames Property

Used to specify the name of the table to request statistics for.

Class

[TIBCStatisticalService](#)

Syntax

```
property TableNames: string;
```

Remarks

Use the TableName property to specify the name of the table for which statistics should be requested.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.18 TIBCTraceService Class

This component is used for working with trace service added in Firebird 2.5.

For a list of all members of this type, see [TIBCTraceService](#) members.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCTraceService = class(TIBCControlAndQueryService);
```

Remarks

The TIBCTraceService component is used to work with trace service added in Firebird 2.5.

Use the TIBCTraceService component to work with trace service. Trace enables various events performed inside the engine, such as statement execution, connections, disconnections, etc., to be logged and collated for real-time analysis of the corresponding performance characteristics.

Trace takes place in the context of a trace session. Each trace session has its own

configuration, state and output. When a user application starts a trace session, it sets SessionName (optional) and the session configuration (Config) (mandatory).

Inheritance Hierarchy

[TCustomIBCSERVICE](#)

[TIBCCONTROLANDQUERYSERVICE](#)

TIBCTRACESERVICE

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.18.1 Members

[TIBCTRACESERVICE](#) class overview.

Properties

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BufferSize (inherited from TIBCCONTROLANDQUERYSERVICE)	Used to set or return the buffer size.
Config	Used to set the configuration of a trace session.
Eof (inherited from TIBCCONTROLANDQUERYSERVICE)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB (inherited from TIBCCONTROLANDQUERYSERVICE)	Used to return and sets SPB parameters.

TCustomIBCSERVICE)	
SessionName	Used to set session name for a trace session.

Methods

Name	Description
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
GetNextChunk (inherited from TIBCCControlAndQueryService)	Returns the next chunk of data.
GetNextLine (inherited from TIBCCControlAndQueryService)	Returns the next line of data.
ListTraceSessions	Returns information about all trace sessions.
ResumeTrace	Resumes a trace session.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.
StartTrace	Starts the trace session.
StopTrace	Stops a trace session.
SuspendTrace	Suspends a trace session.

Events

Name	Description
OnAttach (inherited from TCustomIBCSERVICE)	Occurs when the database is attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.18.2 Properties

Properties of the **TIBCTraceService** class.

For a complete list of the **TIBCTraceService** class members, see the [TIBCTraceService Members](#) topic.

Public

Name	Description
Eof (inherited from TIBControlAndQueryService)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.

Published

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BufferSize (inherited from TIBControlAndQueryService)	Used to set or return the buffer size.
Config	Used to set the configuration of a trace session.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
SessionName	Used to set session name for a trace session.

See Also

- [TIBCTraceService Class](#)

- [TIBCTraceService Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.18.2.1 Config Property

Used to set the configuration of a trace session.

Class

[TIBCTraceService](#)

Syntax

```
property Config: TStrings;
```

Remarks

Use the Config property to set the configuration of a trace session.

The session configuration is a text conforming to the rules and syntax modelled in the fbtrace.conf template that is in Firebird's root directory, apart from the lines relating to placement of the output.

For example, the following configuration is used to trace prepare, free and execution of all statements in the mydatabase.fdb database:

```
<database mydatabase.fdb>
  enabled true
  log_statement_prepare true
  log_statement_free true
  log_statement_start true
  log_statement_finish true
  time_threshold 0
</database>
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.18.2.2 SessionName Property

Used to set session name for a trace session.

Class

[TIBCTraceService](#)

Syntax

```
property SessionName: string;
```

Remarks

Set the SessionName property to distinguish your session in the list of trace sessions.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.18.3 Methods

Methods of the **TIBCTraceService** class.

For a complete list of the **TIBCTraceService** class members, see the [TIBCTraceService Members](#) topic.

Public

Name	Description
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
GetNextChunk (inherited from TIBCCONTROLANDQUERYSERVICE)	Returns the next chunk of data.
GetNextLine (inherited from TIBCCONTROLANDQUERYSERVICE)	Returns the next line of data.
ListTraceSessions	Returns information about all trace sessions.
ResumeTrace	Resumes a trace session.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.
StartTrace	Starts the trace session.
StopTrace	Stops a trace session.
SuspendTrace	Suspends a trace session.

See Also

- [TIBCTraceService Class](#)
- [TIBCTraceService Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.18.3.1 ListTraceSessions Method

Returns information about all trace sessions.

Class

[TIBCTraceService](#)

Syntax

```
procedure ListTraceSessions;
```

Remarks

Call the ListTraceSessions method to get information about all trace sessions. You can read this information from the service output using the GetNextChunk or GetNextLine methods. For each session the service returns a text message in the following format:

- Session ID: <number>
- name: <string>. Prints the trace session name if it is not empty
- user: <string>. Prints the user name of the user that created the trace session
- date: YYYY-MM-DD HH:NN:SS, start date and time of the user session
- flags: <string>

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.18.3.2 ResumeTrace Method

Resumes a trace session.

Class

[TIBCTraceService](#)

Syntax

```
procedure ResumeTrace(SessionID: integer);
```

Parameters

SessionID

Holds a session ID.

Remarks

Call the ResumeTrace method to resume the trace session with the specified ID. You can find the ID of the current trace session in the first lines of its output. Use ListTraceSessions to get ID for all trace sessions.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.18.3.3 StartTrace Method

Starts the trace session.

Class

[TIBCTraceService](#)

Syntax

```
procedure StartTrace;
```

Remarks

Call the StartTrace method to start the trace session. After the session is started you can read its output using the GetNextChunk or GetNextLine methods.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.18.3.4 StopTrace Method

Stops a trace session.

Class

[TIBCTraceService](#)

Syntax

```
procedure StopTrace(SessionID: integer);
```

Parameters

SessionID

Holds a session ID.

Remarks

Call the StopTrace method to stop the trace session with the specified ID. You can find the ID of the current trace session in the first lines of its output. Use ListTraceSessions to get ID for all trace sessions.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.18.3.5 SuspendTrace Method

Suspends a trace session.

Class

[TIBCTraceService](#)

Syntax

```
procedure suspendTrace(SessionID: integer);
```

Parameters

SessionID

Holds a session ID.

Remarks

Call the SuspendTrace method to suspend the trace session with the specified ID. You can find ID of the current trace session in the first lines of its output. Use ListTraceSessions to get ID for all trace sessions.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19 TIBCUserInfo Class

TIBCUserInfo stores information about an InterBase user for the security service.

For a list of all members of this type, see [TIBCUserInfo](#) members.

Unit

[IBAdmin](#)

Syntax

```
TIBCUserInfo = class(TPersistent);
```

Remarks

TIBCUserInfo is the user descriptor that the InterBase security service uses to describe a single user.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.1 Members

[TIBCUserInfo](#) class overview.

Properties

Name	Description
ActiveUser	Used to determine if the user is active.
DefaultRole	Holds the user default role.
Description	Holds the description.
FirstName	Holds the user's first name.
GroupID	Holds the user's group ID.
GroupName	Holds the name of the group.
LastName	Holds the user's Last name.
MiddleName	Holds the user's middle name.
Password	Holds the user's password.

SQLRole	Holds an optional role to use when attaching to the security database.
SystemUserName	Holds the system user name.
UserID	Holds the user's individual ID.
UserName	Holds the user's login name.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2 Properties

Properties of the **TIBCUserInfo** class.

For a complete list of the **TIBCUserInfo** class members, see the [TIBCUserInfo Members](#) topic.

Published

Name	Description
ActiveUser	Used to determine if the user is active.
DefaultRole	Holds the user default role.
Description	Holds the description.
FirstName	Holds the user's first name.
GroupID	Holds the user's group ID.
GroupName	Holds the name of the group.
LastName	Holds the user's Last name.
MiddleName	Holds the user's middle name.
Password	Holds the user's password.
SQLRole	Holds an optional role to use when attaching to the security database.
SystemUserName	Holds the system user name.

UserID	Holds the user's individual ID.
UserName	Holds the user's login name.

See Also

- [TIBCUserInfo Class](#)
- [TIBCUserInfo Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.1 ActiveUser Property

Used to determine if the user is active.

Class

[TIBCUserInfo](#)

Syntax

```
property ActiveUser: Boolean default False;
```

Remarks

Use the ActiveUser property to find out if the user is active.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.2 DefaultRole Property

Holds the user default role.

Class

[TIBCUserInfo](#)

Syntax

```
property DefaultRole: string;
```

Remarks

The DefaultRole property holds the user default role.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.3 Description Property

Holds the description.

Class

[TIBCUserInfo](#)

Syntax

```
property Description: string;
```

Remarks

The Description property holds the description.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.4 FirstName Property

Holds the user's first name.

Class

[TIBCUserInfo](#)

Syntax

```
property FirstName: string;
```

Remarks

The FirstName property holds the user's first name.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.5 GroupID Property

Holds the user's group ID.

Class

[TIBCUserInfo](#)

Syntax

```
property GroupID: Integer default 0;
```

Remarks

The GroupID property holds the user's group ID.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.6 GroupName Property

Holds the name of the group.

Class

[TIBCUserInfo](#)

Syntax

```
property GroupName: string;
```

Remarks

The GroupName property holds the name of the group.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.7 LastName Property

Holds the user's Last name.

Class

[TIBCUserInfo](#)

Syntax

```
property LastName: string;
```

Remarks

The LastName property holds the user's Last name.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.8 MiddleName Property

Holds the user's middle name.

Class

[TIBCUserInfo](#)

Syntax

```
property MiddleName: string;
```

Remarks

The MiddleName property holds the user's middle name.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.9 Password Property

Holds the user's password.

Class

[TIBCUserInfo](#)

Syntax

```
property Password: string;
```

Remarks

The Password property holds the user's password. The password can be maximum 31

characters, only first 8 characters are significant

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.10 SQLRole Property

Holds an optional role to use when attaching to the security database.

Class

[TIBCUserInfo](#)

Syntax

```
property SQLRole: string;
```

Remarks

The SQLRole property is an optional role to use when attaching to the security database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.11 SystemUserName Property

Holds the system user name.

Class

[TIBCUserInfo](#)

Syntax

```
property SystemUserName: string;
```

Remarks

The SystemUserName property holds the system user name.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.12 UserID Property

Holds the user's individual ID.

Class

[TIBCUserInfo](#)

Syntax

```
property UserID: Integer default 0;
```

Remarks

The UserID property holds the the user's individual ID.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.19.2.13 UserName Property

Holds the user's login name.

Class

[TIBCUserInfo](#)

Syntax

```
property UserName: string;
```

Remarks

The UserName property holds the user's login name.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.20 TIBCValidationService Class

Used to validate a database and reconcile database transactions.

For a list of all members of this type, see [TIBCValidationService](#) members.

Unit

[IBCAdmin](#)

Syntax

```
TIBCValidationService = class(TIBCControlAndQueryService);
```

Remarks

Use a TIBCValidationService object to validate a database and reconcile database transactions.

Inheritance Hierarchy

[TCustomIBCSERVICE](#)[TIBCControlAndQueryService](#)**TIBCValidationService**

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.12.1.20.1 Members

[TIBCValidationService](#) class overview.

Properties

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BufferSize (inherited from TIBCControlAndQueryService)	Used to set or return the buffer size.
Database	Used to set or return the database name.
Eof (inherited from TIBCControlAndQueryService)	Used to determine whether the end of the file has been reached.
GlobalAction	Used to set or return the global transaction action.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
LimboTransactionInfo	Used to return the limbo transaction information.

LimboTransactionInfoCount	Used to set or return the number of elements in the TIBCLimboTransactionInfo array.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Options	Used to invoke database validation.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
RecoverTwoPhaseGlobal	Used to restore two-phase transactions.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.

Methods

Name	Description
Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
FetchLimboTransactionInfo	Gets limbo transaction information.
FixLimboTransactionErrors	Fixes limbo transaction errors.
GetNextChunk (inherited from TIBCCControlAndQueryService)	Returns the next chunk of data.
GetNextLine (inherited from TIBCCControlAndQueryService)	Returns the next line of data.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.
SweepDatabase	Performs a database sweep.

Events

Name	Description
OnAttach (inherited from TCustomIBCSERVICE)	Occurs when the database is attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.20.2 Properties

Properties of the **TIBCValidationService** class.

For a complete list of the **TIBCValidationService** class members, see the

[TIBCValidationService Members](#) topic.

Public

Name	Description
Eof (inherited from TIBCControlAndQueryService)	Used to determine whether the end of the file has been reached.
Handle (inherited from TCustomIBCSERVICE)	Used to return the database handle.
LimboTransactionInfo	Used to return the limbo transaction information.
LimboTransactionInfoCount	Used to set or return the number of elements in the TIBCLimboTransactionInfo array.
ServiceParamBySPB (inherited from TCustomIBCSERVICE)	Used to return and sets SPB parameters.

Published

Name	Description
Active (inherited from TCustomIBCSERVICE)	Used to set the service to active or inactive (default).
BufferSize (inherited from TIBCControlAndQueryService)	Used to set or return the buffer size.

Database	Used to set or return the database name.
GlobalAction	Used to set or return the global transaction action.
LoginPrompt (inherited from TCustomIBCSERVICE)	Used to display a login prompt before attaching to a database.
Options	Used to invoke database validation.
Params (inherited from TCustomIBCSERVICE)	Used to set or return database parameters.
Protocol (inherited from TCustomIBCSERVICE)	Used to select the network protocol.
RecoverTwoPhaseGlobal	Used to restore two-phase transactions.
Server (inherited from TCustomIBCSERVICE)	Used to set the name of the server on which the services are to be run.

See Also

- [TIBCValidationService Class](#)
- [TIBCValidationService Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.20.2.1 Database Property

Used to set or return the database name.

Class

[TIBCValidationService](#)

Syntax

```
property Database: string;
```

Remarks

Use the Database property to set or return the database name to set properties on.

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.12.1.20.2.2 GlobalAction Property

Used to set or return the global transaction action.

Class

[TIBValidationService](#)

Syntax

```
property GlobalAction: TIBTransactionGlobalAction default tgNoGlobalAction;
```

Remarks

Use the GlobalAction property to set or return the global transaction action. This value indicates what action to take with the transactions that follow.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.20.2.3 LimboTransactionInfo Property(Indexer)

Used to return the limbo transaction information.

Class

[TIBValidationService](#)

Syntax

```
property LimboTransactionInfo[Index: Integer]: TIBCLimboTransactionInfo;
```

Parameters

Index

Remarks

Use the LimboTransactionInfo property to return the limbo transaction information.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.20.2.4 LimboTransactionInfoCount Property

Used to set or return the number of elements in the TIBCLimboTransactionInfo array.

Class

[TIBCValidationService](#)

Syntax

```
property LimboTransactionInfoCount: Integer;
```

Remarks

Use the LimboTransactionInfoCount property to set or return the number of elements in the TIBCLimboTransactionInfo array.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.20.2.5 Options Property

Used to invoke database validation.

Class

[TIBCValidationService](#)

Syntax

```
property Options: TIBCValidateOptions default [];
```

Remarks

Use the Options property of the TIBCValidationService component to invoke database validation.

Set any of the following options of type `TValidateOption` to `True` to perform the appropriate validation:

Note: Not all combinations of validation options work together. For example, you can not simultaneously mend and validate the database at the same time. Conversely, some options are intended to be used with other options, such as `ValidateDB`, or `ValidateFull` with `ValidateDB`.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.20.2.6 RecoverTwoPhaseGlobal Property

Used to restore two-phase transactions.

Class

[TIBCValidationService](#)

Syntax

```
property RecoverTwoPhaseGlobal: Boolean;
```

Remarks

Use the `RecoverTwoPhaseGlobal` property to restore two-phase transactions.

`RecoverTwoPhaseGlobal` performs automated two-phase recovery, either for a limbo transaction specified by ID or for all limbo transactions.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.20.3 Methods

Methods of the **TIBCValidationService** class.

For a complete list of the **TIBCValidationService** class members, see the [TIBCValidationService Members](#) topic.

Public

Name	Description
------	-------------

Attach (inherited from TCustomIBCSERVICE)	Attaches to the database.
Detach (inherited from TCustomIBCSERVICE)	Detaches from the database.
FetchLimboTransactionInfo	Gets limbo transaction information.
FixLimboTransactionErrors	Fixes limbo transaction errors.
GetNextChunk (inherited from TIBCCControlAndQueryService)	Returns the next chunk of data.
GetNextLine (inherited from TIBCCControlAndQueryService)	Returns the next line of data.
ServiceStart (inherited from TCustomIBCSERVICE)	Starts the service.
SweepDatabase	Performs a database sweep.

See Also

- [TIBCValidationService Class](#)
- [TIBCValidationService Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.20.3.1 FetchLimboTransactionInfo Method

Gets limbo transaction information.

Class

[TIBCValidationService](#)

Syntax

```
procedure FetchLimboTransactionInfo;
```

Remarks

Call the FetchLimboTransactionInfo method to get limbo transaction information from the [LimboTransactionInfo](#) property.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.20.3.2 FixLimboTransactionErrors Method

Fixes limbo transaction errors.

Class

[TIBCValidationService](#)

Syntax

```
procedure FixLimboTransactionErrors;
```

Remarks

Call the FixLimboTransactionErrors method to repair limbo transaction errors.

Note: You should install InterBase 6 to use this feature.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.20.3.3 SweepDatabase Method

Performs a database sweep.

Class

[TIBCValidationService](#)

Syntax

```
procedure SweepDatabase;
```

Remarks

Call the SweepDatabase method to perform a database sweep.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.21 TIBCVersionInfo Class

Represents the version information about an InterBase server.

For a list of all members of this type, see [TIBCVersionInfo](#) members.

Unit

[IBAdmin](#)

Syntax

```
TIBCVersionInfo = class(System.TObject);
```

Remarks

TIBCVersionInfo represents the version information about an InterBase server.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.21.1 Members

[TIBCVersionInfo](#) class overview.

Properties

Name	Description
ServerImplementation	Used to retrieve the implementation string.
ServerVersion	Used to retrieve the version of the InterBase server.
ServiceVersion	Used to retrieve the version number for the service that fetches the version information.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.21.2 Properties

Properties of the **TIBCVersionInfo** class.

For a complete list of the **TIBCVersionInfo** class members, see the [TIBCVersionInfo Members](#) topic.

Public

Name	Description
ServerImplementation	Used to retrieve the implementation string.
ServerVersion	Used to retrieve the version of the InterBase server.
ServiceVersion	Used to retrieve the version number for the service that fetches the version information.

See Also

- [TIBCVersionInfo Class](#)
- [TIBCVersionInfo Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.21.2.1 ServerImplementation Property

Used to retrieve the implementation string.

Class

[TIBCVersionInfo](#)

Syntax

```
property ServerImplementation: string;
```

Remarks

Use the ServerImplementation property to retrieve the implementation string.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.21.2.2 ServerVersion Property

Used to retrieve the version of the InterBase server.

Class

[TIBCVersionInfo](#)

Syntax

```
property ServerVersion: string;
```

Remarks

Use the ServerVersion property to retrieve the version of the InterBase server.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.1.21.2.3 ServiceVersion Property

Used to retrieve the version number for the service that fetches the version information.

Class

[TIBCVersionInfo](#)

Syntax

```
property ServiceVersion: integer;
```

Remarks

Use the ServiceVersion property to retrieve the version number for the service that fetches the version information.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.2 Types

Types in the **IBCAAdmin** unit.

Types

Name	Description
TIBCBackupOptions	Represents the set of TIBCBackupOption .
TIBCNBackupOptions	Represents the set of TIBCNBackupOption .
TIBCRestoreOptions	Represents the set of TIBCRestoreOption .
TIBCStatOptions	Represents the set of TIBCStatOption .
TIBCValidateOptions	Represents the set of TIBCValidateOption .

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.2.1 TIBCBackupOptions Set

Represents the set of [TIBCBackupOption](#).

Unit

[IBCAAdmin](#)

Syntax

```
TIBCBackupOptions = set of TIBCBackupOption;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.2.2 TIBCNBackupOptions Set

Represents the set of [TIBCNBackupOption](#).

Unit

[IBCAAdmin](#)

Syntax

```
TIBCNBackupOptions = set of TIBCNBackupOption;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.2.3 TIBCRestoreOptions Set

Represents the set of [TIBCRestoreOption](#).

Unit

[IBCAdmin](#)

Syntax

```
TIBCRestoreOptions = set of TIBCRestoreOption;
```

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.2.4 TIBCStatOptions Set

Represents the set of [TIBCStatOption](#).

Unit

[IBCAdmin](#)

Syntax

```
TIBCStatOptions = set of TIBCStatOption;
```

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.2.5 TIBCValidateOptions Set

Represents the set of [TIBCValidateOption](#).

Unit

[IBCAdmin](#)

Syntax

```
TIBCValidateOptions = set of TIBCValidateOption;
```

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.3 Enumerations

Enumerations in the **IBCAdmin** unit.

Enumerations

Name	Description
TIBCBackupOption	Allows you to build backup options into your application.
TIBCLicensingAction	Allows to add or remove an InterBase software activation certificate.
TIBCNBackupOption	Options used in TIBCBackupRestoreService for nBackup.
TIBCRestoreOption	Specifies the data restore parameters.
TIBCSecurityAction	Specify the type of the operation for InterBase Security Service to perform.
TIBCStatOption	Allows to specify the way the database statistics would be requested.
TIBCTransactionAdvise	Allows to specify the suggested action for ending the transaction.
TIBCTransactionGlobalAction	Determines which action to take concerning limbo transactions.
TIBCTransactionState	Allows to specify the current state of the limbo transaction.
TIBCValidateOption	Sets the options of validation.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.3.1 TIBCBackupOption Enumeration

Allows you to build backup options into your application.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCBackupOption = (boIgnoreChecksums, boIgnoreLimbo,
boMetadataOnly, boNoGarbageCollection, boOldMetadataDesc,
boNonTransportable, boConvertExtTables);
```

Values

Value	Meaning
boConvertExtTables	Convert external table data to internal tables.
boIgnoreChecksums	Ignore checksums during backup.
boIgnoreLimbo	Ignored limbo transactions during backup.
boMetadataOnly	Output backup file for metadata only with empty tables.
boNoGarbageCollection	Suppress normal garbage collection during backup; improves performance on some databases.
boNonTransportable	Output backup file with non-XDR data format; improves space and performance by a negligible amount.
boOldMetadataDesc	Output metadata in pre-4.0 format.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)**5.12.3.2 TIBCLicensingAction Enumeration**

Allows to add or remove an InterBase software activation certificate.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCLicensingAction = (laAdd, laRemove);
```

Values

Value	Meaning
-------	---------

laAdd	Adds an InterBase software activation certificate.
laRemove	Removes an InterBase software activation certificate.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.3.3 TIBCNBackupOption Enumeration

Options used in [TIBCBackupRestoreService](#) for nBackup.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCNBackupOption = (nboNoTriggers);
```

Values

Value	Meaning
nboNoTriggers	Disables triggers while performing backup or restore.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.3.4 TIBCRestoreOption Enumeration

Specifies the data restore parameters.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCRestoreOption = (roDeactivateIndexes, roNoShadow, roNoValidityCheck, roOneRelationAtATime, roReplace, roCreateNewDB, roUseAllSpace, roValidationCheck);
```

Values

Value	Meaning
-------	---------

roCreateNewDB	Restore but do not overwrite an existing database (restores the database from the backup copy into a new file).
roDeactivateIndexes	Do not build indexes during restore. InterBase rebuilds indexes while restoring a database in the normal mode of operation. If there are duplicates of the records that are unique in the database, the restore process will be terminated. Duplicate records can be added to the database if the unique index is temporary turned off.
roNoShadow	Do not recreate shadow files during restore. In the normal mode of operation InterBase recreates shadow files during restore. By setting this value such option will be blocked.
roNoValidityCheck	Do not enforce validity conditions (for example, NOT NULL) during restore. This value should be used when restoring a database that includes records in a wrong format. If you set this value, the correspondence of data to the existing limitations would not be checked.
roOneRelationAtATime	Commit after completing a restore of each table.
roReplace	Replace database if one exists. If you do not use this value, the operation will be canceled when you try to restore the database.
roUseAllSpace	Do not reserve 20% of each datapage for future record versions; useful for read-only databases.
roValidationCheck	Perform validation check before restoring a database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.3.5 TIBCSecurityAction Enumeration

Specify the type of the operation for InterBase Security Service to perform.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCSecurityAction = (saAddUser, saDeleteUser, saModifyUser,
saDisplayUser);
```

Values

Value	Meaning
-------	---------

saAddUser	New user account will be added.
saDeleteUser	Existing user account will be deleted.
saDisplayUser	Available information on the user account will be displayed.
saModifyUser	The user account will be modified.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.3.6 TIBCStatOption Enumeration

Allows to specify the way the database statistics would be requested.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCStatOption = (soDataPages, soDbLog, soHeaderPages,
soIndexPages, soSystemRelations, soRecordVersions, soStatTables);
```

Values

Value	Meaning
soDataPages	Request statistics for data tables in the database.
soDbLog	Stop reporting statistics after reporting information on the log pages.
soHeaderPages	Stop reporting statistics after reporting the information on the header page.
soIndexPages	Request statistics for the user indexes in the database.
soRecordVersions	Request statistics for the record versions.
soStatTables	Request statistics for tables specified in the TableNames property.
soSystemRelations	Request statistics for system tables and indexes in addition to user tables and indexes.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.3.7 TIBCTransactionAdvise Enumeration

Allows to specify the suggested action for ending the transaction.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCTransactionAdvise = (taCommitAdvise, taRollbackAdvise, taUnknownAdvise);
```

Values

Value	Meaning
taCommitAdvise	Commit the transaction.
taRollbackAdvise	Rollback the transaction.
taUnknownAdvise	No suggested action is specified for ending the transaction.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.3.8 TIBCTransactionGlobalAction Enumeration

Determines which action to take concerning limbo transactions.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCTransactionGlobalAction = (tgNoGlobalAction, tgCommitGlobal, tgRollbackGlobal);
```

Values

Value	Meaning
tgCommitGlobal	Commits the limbo transaction specified by ID or commits all limbo transactions.
tgNoGlobalAction	Takes no action.
tgRollbackGlobal	Rolls back the limbo transaction specified by ID or rolls back all

limbo transactions.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.3.9 TIBCTransactionState Enumeration

Allows to specify the current state of the limbo transaction.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCTransactionState = (tsLimboState, tsCommitState,
tsRollbackState, tsUnknownState);
```

Values

Value	Meaning
tsCommitState	The transaction is in commit state.
tsLimboState	The limbo transaction is in limbo state.
tsRollbackState	The limbo transaction is in rollback state.
tsUnknownState	The state if the limbo transaction is unknown.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.12.3.10 TIBCValidateOption Enumeration

Sets the options of validation.

Unit

[IBCAAdmin](#)

Syntax

```
TIBCValidateOption = (voCheckDB, voIgnoreChecksum, voKillShadows,
voMendDB, voValidateDB, voValidateFull);
```

Values

Value	Meaning
voCheckDB	Request a read-only validation of the database without correcting any problems.
volgnoreChecksum	Ignore all checksum errors when validating or sweeping.
voKillShadows	Remove references to unavailable shadow files.
voMendDB	Mark corrupted records as unavailable so that subsequent operations skip them.
voValidateDB	Locate and release pages that are allocated but unassigned to any data structures.
voValidateFull	Check record and page structures, releasing unassigned record fragments; use with <i>ValidateDB</i> .

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13 IBCAlerter

This unit contains implementation of the TIBCAlerter component.

Classes

Name	Description
TIBCAlerter	A component for transferring messages between connections.

Types

Name	Description
TIBCAlerterEvent	This type is used for the TIBCAlerter.OnEvent event.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13.1 Classes

Classes in the **IBCAlerter** unit.

Classes

Name	Description
TIBCAlerter	A component for transferring messages between connections.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13.1.1 TIBCAlerter Class

A component for transferring messages between connections.

For a list of all members of this type, see [TIBCAlerter](#) members.

Unit

[IBCAlerter](#)

Syntax

```
TIBCAlerter = class(TDAAlerter);
```

Remarks

The TIBCAlerter component allows you to register interest in and asynchronously handle events posted by an InterBase server. Use TIBCAlerter to handle events for responding to actions and database changes made by other applications. To get events application must register required events. To do it set the [TIBCAlerter.Events](#) property to required events and call M:Devart.IbDac.TIBCAlerter.Start() method. When one of the registered events occurs [TIBCAlerter.OnEvent](#) handler is called.

Note: Events are transaction-based. This means that the waiting connection does not get event until the transaction posting the event commits.

Inheritance Hierarchy

[TDAAlerter](#)

TIBCAlerter

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13.1.1.1 Members

[TIBCAlerter](#) class overview.

Properties

Name	Description
Active (inherited from TDAAlerter)	Used to determine if TDAAlerter waits for messages.
AutoCommit	Used to automatically commit transaction after calling the SendEvent method.
AutoRegister (inherited from TDAAlerter)	Used to automatically register events whenever connection opens.
Connection	Used to specify the connection for TIBCAlerter.
Events	Used to specify the names of events to wait.
Transaction	Used to set the transaction to be used by the SendEvent method.

Methods

Name	Description
SendEvent	Sends an event with Name.
Start (inherited from TDAAlerter)	Starts waiting process.
Stop (inherited from TDAAlerter)	Stops waiting process.

Events

Name	Description
OnError (inherited from TDAAlerter)	Occurs if an exception occurs in waiting process
OnEvent	Occurs when waiting process receives an event from the InterBase server.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13.1.1.2 Properties

Properties of the **TIBCAlerter** class.

For a complete list of the **TIBCAlerter** class members, see the [TIBCAlerter Members](#) topic.

Public

Name	Description
Active (inherited from TDAAlerter)	Used to determine if TDAAlerter waits for messages.
AutoRegister (inherited from TDAAlerter)	Used to automatically register events whenever connection opens.

Published

Name	Description
AutoCommit	Used to automatically commit transaction after calling the SendEvent method.
Connection	Used to specify the connection for TIBCAlerter.
Events	Used to specify the names of events to wait.
Transaction	Used to set the transaction to be used by the SendEvent method.

See Also

- [TIBCAlerter Class](#)
- [TIBCAlerter Class Members](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13.1.1.2.1 AutoCommit Property

Used to automatically commit transaction after calling the SendEvent method.

Class

[TIBCAlerter](#)

Syntax

```
property AutoCommit: boolean;
```

Remarks

Use the AutoCommit property to automatically commit transaction after calling the SendEvent method. Events are transaction-based. This means that the waiting connection does not get event until the transaction posting the event commits.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13.1.1.2.2 Connection Property

Used to specify the connection for TIBCAlerter.

Class

[TIBCAlerter](#)

Syntax

```
property Connection: TIBConnection;
```

Remarks

Use the Connection property to specify the connection for TIBCAlerter.

See Also

- [TIBConnection](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13.1.1.2.3 Events Property

Used to specify the names of events to wait.

Class

[TIBCAlerter](#)

Syntax

```
property Events: TStrings;
```

Remarks

Use the Events property to set the names of events to wait.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13.1.1.2.4 Transaction Property

Used to set the transaction to be used by the SendEvent method.

Class

[TIBCAlerter](#)

Syntax

```
property Transaction: TIBCTransaction stored IsTransactionStored;
```

Remarks

Use the Transaction property to set the transaction to be used by the SendEvent method.

Events are transaction-based. This means that the waiting connection does not get event until the transaction posting the event commits.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13.1.1.3 Methods

Methods of the **TIBCAlerter** class.

For a complete list of the **TIBCAlerter** class members, see the [TIBCAlerter Members](#) topic.

Public

Name	Description
SendEvent	Sends an event with Name.
Start (inherited from TDAAlerter)	Starts waiting process.
Stop (inherited from TDAAlerter)	Stops waiting process.

See Also

- [TIBCAlerter Class](#)
- [TIBCAlerter Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.13.1.1.3.1 SendEvent Method

Sends an event with Name.

Class

[TIBCAlerter](#)

Syntax

```
procedure SendEvent(const Name: string);
```

Parameters

Name

Holds the name of the event to send.

Remarks

Call the SendEvent procedure to send an event with Name.

This procedure is supported only for Firebird 2.0 and higher.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.13.1.1.4 Events

Events of the **TIBCAlerter** class.

For a complete list of the **TIBCAlerter** class members, see the [TIBCAlerter Members](#) topic.

Public

Name	Description
OnError (inherited from TDAAlert)	Occurs if an exception occurs in waiting process

Published

Name	Description
OnEvent	Occurs when waiting process receives an event from the InterBase server.

See Also

- [TIBCAlerter Class](#)
- [TIBCAlerter Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13.1.1.4.1 OnEvent Event

Occurs when waiting process receives an event from the InterBase server.

Class

[TIBCAlerter](#)

Syntax

```
property OnEvent: TIBCAlerterEvent;
```

Remarks

The OnEvent event occurs when waiting process receives event from the InterBase server.

The EventName parameter contains the name of the event and the EventCount parameter

contains the number of events that were raised in the transaction. Waiting connection does not get event until the transaction signaling the alert commits and there can be several event raises in one transaction.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13.2 Types

Types in the **IBCAlerter** unit.

Types

Name	Description
TIBCAlerterEvent	This type is used for the TIBCAlerter.OnEvent event.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.13.2.1 TIBCAlerterEvent Procedure Reference

This type is used for the [TIBCAlerter.OnEvent](#) event.

Unit

[IBCAlerter](#)

Syntax

```
TIBCAlerterEvent = procedure (Sender: TObject; EventName: string;
EventCount: Integer) of object;
```

Parameters

Sender

An object that raised the event.

EventName

Holds the name of the event.

EventCount

Holds the number of events that were raised in the transaction.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.14 IBCArray

This unit contains the TIBCArray class for working the InterBase/Firebird array data types.

Classes

Name	Description
TCustomIBCArray	A base class representing the value of the InterBase array data type.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1 Classes

Classes in the **IBCArray** unit.

Classes

Name	Description
TCustomIBCArray	A base class representing the value of the InterBase array data type.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1 TCustomIBCArray Class

A base class representing the value of the InterBase array data type.

For a list of all members of this type, see [TCustomIBCArray](#) members.

Unit

[IBCArray](#)

Syntax

```
TCustomIBCArray = class(TDBObject);
```

Inheritance Hierarchy

[TSharedObject](#)

[TDBObject](#)

TCustomBCArray

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.1 Members

[TCustomBCArray](#) class overview.

Properties

Name	Description
ArrayDimensions	Contains the array dimensions count.
ArrayHighBound	Used to get or set the upper boundary of the defined dimension subscript.
ArrayID	Contains the array ID.
ArrayLowBound	Used to get or set the lower boundary of the defined dimension subscript.
ArraySize	Used to determine the size of the whole array data in bytes.
AsString	Used to return array as string.
Cached	Indicates whether to cache array data on the client side.
CachedDimensions	Contains cached array dimensions count.
CachedHighBound	Used to get the upper boundary of defined dimension subscript of cached array elements.
CachedLowBound	Used to get the lower boundary of the defined dimension subscript of cached array elements.

CachedSize	Used to get the cached array data size in bytes.
ColumnName	Used to get or set the name of the table column that has array type.
DbHandle	Contains the handle of a database where the array is stored.
IsNull	Used to define whether the array field in the database is null.
Items	Used to get or set array items.
ItemScale	Used to get or set the scale for array items for the NUMERIC and DECIMAL datatypes.
ItemSize	Contains the size of an array item.
Modified	Used to indicate if the modifications done in cache were saved to the database.
RefCount (inherited from TSharedObject)	Used to return the count of reference to a TSharedObject object.
TableName	Used to set the name of the table containing an array field.
TrHandle	Contains the handle of the transaction in which the array is read or written.

Methods

Name	Description
AddRef (inherited from TSharedObject)	Increments the reference count for the number of references dependent on the TSharedObject object.
Assign	Copies the Source object content to the current one.
ClearArray	Clears all array values on the server if Cached is set to False.

CreateTemporaryArray	Creates a temporary array on the InterBase server.
GetArrayInfo	Used to get array descriptor.
GetItemAsDateTime	Reads the value of an array item into an object or variable of the TDateTime type.
GetItemAsFloat	Reads the value of an array item into a floating-point number.
GetItemAsInteger	Reads the value of an array item into a integer.
GetItemAsSmallInt	Reads the value of an array item into a short integer.
GetItemAsString	Reads the value of an array item into a string.
GetItemAsWideString	Reads the value of an array item into a WideString.
GetItemsSlice	Returns the array slice items' values.
GetItemValue	Returns the array item value.
ReadArray	Reads an array from the database to memory.
ReadArrayItem	Reads the array item specified by indices from the database to memory.
ReadArraySlice	Reads array slice from the database to memory.
Release (inherited from TSharedObject)	Decrements the reference count.
SetItemAsDateTime	Assigns the TDateTime value to the contents of an array item.
SetItemAsFloat	Assigns floating-point value to the contents of an array item.
SetItemAsInteger	Assigns integer value to the contents of an array item.
SetItemAsSmallInt	Assigns short integer value to the contents of an array item.
SetItemAsString	Assigns string value to the contents of an array item.

SetItemAsWideString	Assigns WideString value to the contents of an array item.
SetItemsSlice	Sets the array slice values.
SetItemValue	Sets the array item value.
WriteArray	Writes all cached array values to the database.
WriteArraySlice	Writes cached array slice.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2 Properties

Properties of the **TCustomIBCArray** class.

For a complete list of the **TCustomIBCArray** class members, see the [TCustomIBCArray Members](#) topic.

Public

Name	Description
ArrayDimensions	Contains the array dimensions count.
ArrayHighBound	Used to get or set the upper boundary of the defined dimension subscript.
ArrayID	Contains the array ID.
ArrayLowBound	Used to get or set the lower boundary of the defined dimension subscript.
ArraySize	Used to determine the size of the whole array data in bytes.
AsString	Used to return array as string.
Cached	Indicates whether to cache array data on the client side.
CachedDimensions	Contains cached array dimensions count.
CachedHighBound	Used to get the upper boundary of defined

	dimension subscript of cached array elements.
CachedLowBound	Used to get the lower boundary of the defined dimension subscript of cached array elements.
CachedSize	Used to get the cached array data size in bytes.
ColumnName	Used to get or set the name of the table column that has array type.
DbHandle	Contains the handle of a database where the array is stored.
IsNull	Used to define whether the array field in the database is null.
Items	Used to get or set array items.
ItemScale	Used to get or set the scale for array items for the NUMERIC and DECIMAL datatypes.
ItemSize	Contains the size of an array item.
Modified	Used to indicate if the modifications done in cache were saved to the database.
RefCount (inherited from TSharedObject)	Used to return the count of reference to a TSharedObject object.
TableName	Used to set the name of the table containing an array field.
TrHandle	Contains the handle of the transaction in which the array is read or written.

See Also

- [TCustomIBCArrary Class](#)
- [TCustomIBCArrary Class Members](#)

Devart. All Rights Reserved.

5.14.1.1.2.1 ArrayDimensions Property

Contains the array dimensions count.

Class

[TCustomIBCArrary](#)

Syntax

```
property ArrayDimensions: integer;
```

Remarks

The ArrayDimensions property is used to hold the array dimensions count. InterBase supports multi-dimensional arrays with 1 to 16 dimensions.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.2 ArrayHighBound Property(Indexer)

Used to get or set the upper boundary of the defined dimension subscript.

Class

[TCustomIBCArrary](#)

Syntax

```
property ArrayHighBound[Dimension: integer]: integer;
```

Parameters

Dimension

Holds the dimension subscript.

Remarks

Use the ArrayHighBound property to get or set the upper boundary of the defined dimension subscript.

See Also

- [ArrayLowBound](#)
- [ArrayDimensions](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.3 ArrayID Property

Contains the array ID.

Class

[TCustomIBCArray](#)

Syntax

```
property ArrayID: TISC_QUAD;
```

Remarks

The ArrayID property is used to hold the array ID that is actually stored in the array field of the table record. It is a unique numeric value that references the array data.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.4 ArrayLow Bound Property(Indexer)

Used to get or set the lower boundary of the defined dimension subscript.

Class

[TCustomIBCArray](#)

Syntax

```
property ArrayLowBound[Dimension: integer]: integer;
```

Parameters

Dimension

Holds the dimension subscript.

Remarks

Use the `ArrayLowBound` property to get or set the lower boundary of the defined dimension subscript.

See Also

- [ArrayHighBound](#)
- [ArrayDimensions](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.5 ArraySize Property

Used to determine the size of the whole array data in bytes.

Class

[TCustomIBCArray](#)

Syntax

```
property ArraySize: integer;
```

Remarks

Use the `ArraySize` property to find out the size of the whole array data in bytes.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.6 AsString Property

Used to return array as string.

Class

[TCustomIBCArray](#)

Syntax

```
property AsString: string;
```

Remarks

Use the AsString property to return array as string. For example, AsString property for two-dimensional array of integer with 2 rows, 3 elements in width can have the following value:

'((1; 2; 3), (4; 5; 6))'. Array values can be set using this property.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.7 Cached Property

Indicates whether to cache array data on the client side.

Class

[TCustomIBCArrary](#)

Syntax

```
property cached: boolean;
```

Remarks

The Cached property is used to define whether to cache array data on the client side.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.8 CachedDimensions Property

Contains cached array dimensions count.

Class

[TCustomIBCArrary](#)

Syntax

```
property cachedDimensions: integer;
```

Remarks

The CachedDimensions property is used to hold the cached array dimensions count.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.14.1.1.2.9 CachedHighBound Property(Indexer)

Used to get the upper boundary of defined dimension subscript of cached array elements.

Class

[TCustomIBCArrary](#)

Syntax

```
property CachedHighBound[Dimension: integer]: integer;
```

Parameters

Dimension

Holds the dimension subscript.

Remarks

Use the CachedHighBound property to get the upper boundary of defined dimension subscript of cached array elements.

See Also

- [ArrayHighBound](#)
- [CachedLowBound](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.10 CachedLow Bound Property(Indexer)

Used to get the lower boundary of the defined dimension subscript of cached array elements.

Class

[TCustomIBCArrary](#)

Syntax

```
property CachedLowBound[Dimension: integer]: integer;
```

Parameters

Dimension

Holds the dimension subscript.

Remarks

Use `CachedLowBound` property to get the lower boundary of the defined dimension subscript of cached array elements.

See Also

- [ArrayLowBound](#)
- [CachedHighBound](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.11 `CachedSize` Property

Used to get the cached array data size in bytes.

Class

[TCustomIBCArrary](#)

Syntax

```
property CachedSize: integer;
```

Remarks

Use the `CachedSize` property to get the cached array data size in bytes.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.12 `ColumnName` Property

Used to get or set the name of the table column that has array type.

Class

[TCustomIBCArrary](#)

Syntax

```
property ColumnName: string;
```

Remarks

Use the ColumnName property to get or set the name of the table column that has array type.

See Also

- [GetArrayInfo](#)
- [TableName](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.13 DbHandle Property

Contains the handle of a database where the array is stored.

Class

[TCustomIBCArray](#)

Syntax

```
property DbHandle: TISC_DB_HANDLE;
```

Remarks

The DbHandle property is used to hold the handle of a database where the array is stored.

See Also

- [TIBCConnection.Handle](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.14 IsNull Property

Used to define whether the array field in the database is null.

Class

[TCustomIBCArray](#)

Syntax

```
property IsNull: boolean;
```

Remarks

Use the IsNull property to define whether the array field in the database is null.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.15 Items Property

Used to get or set array items.

Class

[TCustomIBCArrary](#)

Syntax

```
property Items: variant;
```

Remarks

Returns varArray, containing array items. Use the Items property to get or set array items.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.16 ItemScale Property

Used to get or set the scale for array items for the NUMERIC and DECIMAL datatypes.

Class

[TCustomIBCArrary](#)

Syntax

```
property ItemScale: integer;
```

Remarks

Use the ItemScale property to get or set the scale for array items for the NUMERIC and DECIMAL datatypes.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.17 ItemSize Property

Contains the size of an array item.

Class

[TCustomIBCArrary](#)

Syntax

```
property ItemSize: integer;
```

Remarks

The ItemSize property is used to hold the size of an array item in bytes.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.18 Modified Property

Used to indicate if the modifications done in cache were saved to the database.

Class

[TCustomIBCArrary](#)

Syntax

```
property Modified: boolean;
```

Remarks

The Modified property is True when the array was modified in cache and these changes were not saved to the database.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.19 TableName Property

Used to set the name of the table containing an array field.

Class

[TCustomIBCArray](#)

Syntax

```
property TableName: string;
```

Remarks

Use the TableName property to get or set the name of the table containing an array field.

See Also

- [GetArrayInfo](#)
- [ColumnName](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.2.20 TrHandle Property

Contains the handle of the transaction in which the array is read or written.

Class

[TCustomIBCArray](#)

Syntax

```
property TrHandle: TISC_TR_HANDLE;
```

Remarks

The TrHandle property is used to hold the handle of the transaction in which the array is read or written.

See Also

- [TIBCTransaction.Handle](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3 Methods

Methods of the **TCustomIBCArry** class.

For a complete list of the **TCustomIBCArry** class members, see the [TCustomIBCArry Members](#) topic.

Public

Name	Description
AddRef (inherited from TSharedObject)	Increments the reference count for the number of references dependent on the TSharedObject object.
Assign	Copies the Source object content to the current one.
ClearArray	Clears all array values on the server if Cached is set to False.
CreateTemporaryArray	Creates a temporary array on the InterBase server.
GetArrayInfo	Used to get array descriptor.
GetItemAsDateTime	Reads the value of an array item into an object or variable of the TDateTime type.
GetItemAsFloat	Reads the value of an array item into a floating-point number.
GetItemAsInteger	Reads the value of an array item into a integer.
GetItemAsSmallInt	Reads the value of an array item into a short integer.
GetItemAsString	Reads the value of an array item into a string.
GetItemAsWideString	Reads the value of an array item into a WideString.
GetItemsSlice	Returns the array slice items' values.

GetValue	Returns the array item value.
ReadArray	Reads an array from the database to memory.
ReadArrayItem	Reads the array item specified by indices from the database to memory.
ReadArraySlice	Reads array slice from the database to memory.
Release (inherited from TSharedObject)	Decrements the reference count.
SetItemAsDateTime	Assigns the TDateTime value to the contents of an array item.
SetItemAsFloat	Assigns floating-point value to the contents of an array item.
SetItemAsInteger	Assigns integer value to the contents of an array item.
SetItemAsSmallInt	Assigns short integer value to the contents of an array item.
SetItemAsString	Assigns string value to the contents of an array item.
SetItemAsWideString	Assigns WideString value to the contents of an array item.
SetItemsSlice	Sets the array slice values.
SetItemValue	Sets the array item value.
WriteArray	Writes all cached array values to the database.
WriteArraySlice	Writes cached array slice.

See Also

- [TCustomIBCArrary Class](#)
- [TCustomIBCArrary Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.1 Assign Method

Copies the Source object content to the current one.

Class

[TCustomIBCArray](#)

Syntax

```
procedure Assign(Source: TCustomIBCArray);
```

Parameters

Source

Holds the Source object content.

Remarks

Use the Assign method to copy the Source object content to the current one.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.2 ClearArray Method

Clears all array values on the server if Cached is set to False.

Class

[TCustomIBCArray](#)

Syntax

```
procedure ClearArray;
```

Remarks

Call the ClearArray method to clear all array values on the server if Cached is set to False. If Cached property is set to True, this method clears all cached array values.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.3 CreateTemporaryArray Method

Creates a temporary array on the InterBase server.

Class

[TCustomIBCArrary](#)

Syntax

```
procedure CreateTemporaryArray;
```

Remarks

Call the CreateTemporaryArray method to create a temporary array on the InterBase server. To use this method, the ArrayDimensions, ItemType, ItemSize, ItemScale properties must be set.

See Also

- [GetArrayInfo](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.4 GetArrayInfo Method

Used to get array descriptor.

Class

[TCustomIBCArrary](#)

Syntax

```
procedure GetArrayInfo;
```

Remarks

Call the GetArrayInfo method to get array descriptor, which contains information about array dimensions, high and low subscript boundaries, array item datatype, size and scale. This method is useful when using TIBCArrary as IN parameter. It can be also useful for creating temporary array with the same array dimensions and item type as array field in a table. For doing this set TableName and ColumnName properties for TIBCArrary object, call GetArrayInfo

method and then call `CreateTemporaryArray` method.

See Also

- [CreateTemporaryArray](#)
- [TableName](#)
- [ColumnName](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.5 GetItemAsDateTime Method

Reads the value of an array item into an object or variable of the `TDateTime` type.

Class

[TCustomIBCArray](#)

Syntax

```
function GetItemAsDateTime(const Indices: array of integer):  
TDateTime;
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Return Value

the value of an array item as `DateTime`.

Remarks

Call the `GetItemAsDateTime` method to read the value of an array item into an object or variable of the `TDateTime` type.

See Also

- [GetItemValue](#)
- [TIBCArray.ItemType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.14.1.1.3.6 GetItemAsFloat Method

Reads the value of an array item into a floating-point number.

Class

[TCustomIBArray](#)

Syntax

```
function GetItemAsFloat(const Indices: array of integer): double;
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Return Value

the value of an array item as a floating-point number.

Remarks

Call the GetItemAsFloat method to read the value of an array item into a floating-point number.

See Also

- [GetItemValue](#)
- [TIBArray.ItemType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.7 GetItemAsInteger Method

Reads the value of an array item into a integer.

Class

[TCustomIBArray](#)

Syntax

```
function GetItemAsInteger(const Indices: array of integer):
```

```
integer;
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Return Value

the value of an array item as integer.

Remarks

Call the `GetItemAsInteger` method to read the value of an array item into a integer.

See Also

- [GetItemValue](#)
- [TIBCArray.ItemType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.8 GetItemAsSmallInt Method

Reads the value of an array item into a short integer.

Class

[TCustomIBCArray](#)

Syntax

```
function GetItemAsSmallInt(const Indices: array of integer):  
SmallInt;
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Return Value

the value of an array item as short integer.

Remarks

Call the `GetItemAsSmallInt` method to read the value of an array item into a short integer.

See Also

- [GetItemValue](#)
- [TIBCArray.ItemType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.9 GetItemAsString Method

Reads the value of an array item into a string.

Class

[TCustomIBCArray](#)

Syntax

```
function GetItemAsString(const Indices: array of integer):  
string;
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Return Value

the value of an array item as string.

Remarks

Call the GetItemAsString method to read the value of an array item into a string.

See Also

- [GetItemValue](#)
- [TIBCArray.ItemType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.10 GetItemAsWideString Method

Reads the value of an array item into a WideString.

Class

[TCustomIBCArrary](#)

Syntax

```
function GetItemAsWideString(const Indices: array of integer):  
string;
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Return Value

the value of an array item as WideString.

Remarks

Call the GetItemAsWideString method to read the value of an array item into a WideString.

See Also

- [GetItemValue](#)
- [TIBCArrary.ItemType](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.11 GetItemsSlice Method

Returns the array slice items' values.

Class

[TCustomIBCArrary](#)

Syntax

```
function GetItemsSlice(const Bounds: array of integer): variant;
```

Parameters

Bounds

Holds the upper and lower boundaries of the slice.

Return Value

the array slice items' values.

Remarks

Call the GetItemsSlice method to get the array slice items' values.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.12 GetItemValue Method

Returns the array item value.

Class

[TCustomIBCArrary](#)

Syntax

```
function GetItemValue(const Indices: array of integer): variant;
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Return Value

the value of an array item.

Remarks

Call the GetItemValue method to get the array item value.

See Also

- [GetItemAsDateTime](#)
- [GetItemAsFloat](#)
- [GetItemAsInteger](#)
- [GetItemAsSmallInt](#)
- [GetItemAsString](#)

- [GetItemAsWideString](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.13 ReadArray Method

Reads an array from the database to memory.

Class

[TCustomIBCArrary](#)

Syntax

```
procedure ReadArray;
```

Remarks

Call the ReadArray method to read an array from database to memory.

See Also

- [WriteArray](#)
- [ReadArraySlice](#)
- [ReadArrayItem](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.14 ReadArrayItem Method

Reads the array item specified by indices from the database to memory.

Class

[TCustomIBCArrary](#)

Syntax

```
procedure ReadArrayItem(const Indices: array of integer);
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Remarks

Call the `ReadArrayItem` method to read array item specified by indices from the database to the memory.

See Also

- [ReadArray](#)
- [ReadArraySlice](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.15 ReadArraySlice Method

Reads array slice from the database to memory.

Class

[TCustomIBCArrary](#)

Syntax

```
procedure ReadArraySlice(const Bounds: array of integer);
```

Parameters

Bounds

Holds the upper and lower boundaries of the array slice.

Remarks

Call the `ReadArraySlice` method to read array slice from database to memory.

See Also

- [WriteArraySlice](#)
- [ReadArray](#)
- [ReadArrayItem](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.14.1.1.3.16 SetItemAsDateTime Method

Assigns the TDateTime value to the contents of an array item.

Class

[TCustomIBCArray](#)

Syntax

```
procedure SetItemAsDateTime(const Indices: array of integer;  
const Value: TDateTime);
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Value

Holds the array item value as TDateTime.

Remarks

Call the SetItemAsDateTime method to assign the TDateTime value to the contents of an array item.

See Also

- [SetItemValue](#)
- [TIBCArray.ItemType](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.17 SetItemAsFloat Method

Assigns floating-point value to the contents of an array item.

Class

[TCustomIBCArray](#)

Syntax

```
procedure SetItemAsFloat(const Indices: array of integer; Value: double);
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Value

Holds the array item value as floating-point.

Remarks

Call the SetItemAsFloat method to assign floating-point value to the contents of an array item.

See Also

- [SetItemValue](#)
- [TIBCArray.ItemType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.18 SetItemAsInteger Method

Assigns integer value to the contents of an array item.

Class

[TCustomIBCArray](#)

Syntax

```
procedure SetItemAsInteger(const Indices: array of integer; Value: integer);
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Value

Holds the array item value as integer.

Remarks

Call the `SetItemAsInteger` method to assign integer value to the contents of an array item.

See Also

- [SetItemValue](#)
- [TIBCArry.ItemType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.19 SetItemAsSmallInt Method

Assigns short integer value to the contents of an array item.

Class

[TCustomIBCArry](#)

Syntax

```
procedure SetItemAsSmallInt(const Indices: array of integer;  
value: SmallInt);
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Value

Holds the array item value as short integer.

Remarks

Call the `SetItemAsSmallInt` method to assign short integer value to the contents of an array item.

See Also

- [SetItemValue](#)
- [TIBCArry.ItemType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.20 SetItemAsString Method

Assigns string value to the contents of an array item.

Class

[TCustomIBCArray](#)

Syntax

```
procedure SetItemAsString(const Indices: array of integer; const
value: string);
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Value

Holds the array item value as string.

Remarks

Call the SetItemAsString method to assign string value to the contents of an array item.

See Also

- [SetItemValue](#)
- [TIBCArray.ItemType](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.21 SetItemAsWideString Method

Assigns WideString value to the contents of an array item.

Class

[TCustomIBCArray](#)

Syntax

```
procedure SetItemAsWideString(const Indices: array of integer;
const value: string);
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Value

Holds the array item value as WideString.

Remarks

Call the `SetItemAsWideString` method to assign WideString value to the contents of an array item.

See Also

- [SetItemValue](#)
- [TIBCArry.ItemType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.22 SetItemsSlice Method

Sets the array slice values.

Class

[TCustomIBCArry](#)

Syntax

```
procedure SetItemsSlice(const values: variant);
```

Parameters

Values

Holds the array slice values.

Remarks

Call the `SetItemsSlice` method to set array slice values.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.23 SetItemValue Method

Sets the array item value.

Class

[TCustomIBCArray](#)

Syntax

```
procedure SetItemValue(const Indices: array of integer; const  
value: variant);
```

Parameters

Indices

Holds an array of element indexes in the array (if the array is one-dimensional, there is only one index).

Value

Holds the array item value.

Remarks

Call the SetItemValue method to set the array item value.

See Also

- [SetItemAsDateTime](#)
- [SetItemAsFloat](#)
- [SetItemAsInteger](#)
- [SetItemAsSmallInt](#)
- [SetItemAsString](#)
- [SetItemAsWideString](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.24 WriteArray Method

Writes all cached array values to the database.

Class

[TCustomIBCArray](#)

Syntax

```
procedure writeArray;
```

Remarks

Call the WriteArray method to write all cached array values to the database. This method does nothing if the Cached property is set to False.

See Also

- [WriteArraySlice](#)
- [ReadArray](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.14.1.1.3.25 WriteArraySlice Method

Writes cached array slice.

Class

[TCustomIBCArray](#)

Syntax

```
procedure writeArraySlice(const Bounds: array of integer);
```

Parameters

Bounds

Holds the upper and lower boundaries of the array slice.

Remarks

Call the WriteArraySlice method to write cached array slice. This method does nothing if Cached property is set to False.

See Also

- [WriteArray](#)

- [ReadArraySlice](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15 IBCClasses

IBCClasses unit defines the following data type constants: dtDbKey dtFixedChar dtFixedWideChar

Classes

Name	Description
TGDSDatabaseInfo	A class providing information about an InterBase database.
TIBCBlob	A class holding value of the BLOB fields and parameters.

Enumerations

Name	Description
TIBCIsolationLevel	Specifies the transaction isolation level and access mode.

Routines

Name	Description
Reverse2	Switches places of bytes in the word argument.
Reverse4	Switches places of bytes in the cardinal argument.
XSQLDA_LENGTH	Analogue of InterBase XSQLDA_LENGTH macro. Calculates the number of bytes that must be allocated for an input or output XSQLDA.

Variables

Name	Description
IntegerPrecision	Set this constant to define the type of NUMERIC and DECIMAL fields with precision less or equal than IntegerPrecision as dtInteger. Otherwise, they are defined as dtFloat.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1 Classes

Classes in the **IBCClasses** unit.

Classes

Name	Description
TGDSDatabaseInfo	A class providing information about an InterBase database.
TIBCBlob	A class holding value of the BLOB fields and parameters.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1 TGDSDatabaseInfo Class

A class providing information about an InterBase database.

For a list of all members of this type, see [TGDSDatabaseInfo](#) members.

Unit

[IBCClasses](#)

Syntax

```
TGDSDatabaseInfo = class(System.TObject);
```

Remarks

Use the `TGDSDatabaseInfo` class to get information about InterBase database.

See Also

- [TIBCConnection.DatabaseInfo](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.1 Members

[TGSDatabaseInfo](#) class overview.

Properties

Name	Description
Allocation	Indicates the number of database pages allocated.
AttachmentID	Indicates a unique connection identifier.
BackoutCount	Contains the number of removals of a record version.
BaseLevel	Indicates the database version number.
CurLogFileName	Contains the name of the log-file.
CurLogPartitionOffset	Indicates the value of <code>isc_info_cur_log_part_offset</code> .
CurrentMemory	Indicates the amount of the server memory currently in use.
DBFileName	Contains the database filename.
DBImplementationClass	Indicates the database implementation class number.
DBImplementationNo	Indicates the database implementation number.
DBSiteName	Contains the database site name.

DeleteCount	Holds the number of database deletes since the database was last attached.
ExpungeCount	Indicates the number of removals of a record and all of its ancestors.
Fetches	Indicates the number of reads from the memory buffer cache.
ForcedWrites	Used to indicate the mode in which database writes are performed.
InsertCount	Holds the number of insertions into the database since the database was last attached.
IsRemoteConnect	Used to indicate whether the connection with the database is remote.
LogFile	Used to indicate the value of isc info log file.
Marks	Used to indicate the number of writes to the memory buffer cache.
MaxMemory	Indicates the maximum amount of memory used at one time since the first process attached to the database.
NoReserve	Specifies if the space for holding backup versions of modified records is reserved on each database page.
NumBuffers	Indicates the number of currently allocated memory buffers.
NumWALBuffers	Indicates the value of isc info num wal buffers.
ODSMajorVersion	Indicates the on disk structure (ODS) major version number.
ODSMinorVersion	Indicates the on disk structure (ODS) minor

	version number.
PageSize	Shows the number of bytes per page for the database.
PurgeCount	Holds the number of removals of records committed from the database, resulting in older versions.
ReadIdxCount	Holds the number of reads done via an index since the database was last attached.
ReadOnly	Indicates whether the database is read only.
Reads	Indicates the number of page reads from the database since the current database was first attached.
ReadSeqCount	Contains the number of sequential database reads done on each table since the database was last attached.
SweepInterval	Indicates the number of transactions that are committed between "sweeps".
UpdateCount	Contains the number of database updates since the database was last attached.
UserNames	Contains the names of all users currently attached to the database.
Version	Indicates the version of the database implementation.
WALAverageGroupCommitSize	Indicates the value of <code>isc_info_wal_avg_grpc_size</code> .
WALAverageIOSize	Indicates the value of <code>isc_info_wal_avg_io_size</code> .
WALBufferSize	Indicates the value of <code>isc_info_wal_buffer_size</code> .
WALCheckpointLength	Indicates the value of <code>isc_info_wal_ckpt_length</code> .
WALCurCheckpointInterval	Indicates the value of <code>isc_info_wal_cur_ckpt_inter</code> .

	val.
WALGroupCommitWaitUSecs	Indicates the value of isc_info_wal_grpc_wait_usecs.
WALNumCommits	Indicates the value of isc_info_wal_num_commits.
WALNumIO	Indicates the value of isc_info_wal_num_id.
WALPrvCheckpointFilename	Indicates the value of isc_info_wal_prv_ckpt_filename.
WALPrvCheckpointPartOffset	Indicates the value of isc_info_wal_prv_ckpt_offset.
Writes	Indicates the number of page writes to the current database since it was first attached by any process.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2 Properties

Properties of the **TGDSDatabaseInfo** class.

For a complete list of the **TGDSDatabaseInfo** class members, see the [TGDSDatabaseInfo Members](#) topic.

Public

Name	Description
Allocation	Indicates the number of database pages allocated.
AttachmentID	Indicates a unique connection identifier.
BackoutCount	Contains the number of removals of a record version.
BaseLevel	Indicates the database version number.
CurLogFileName	Contains the name of the log-file.

CurLogPartitionOffset	Indicates the value of isc_info_cur_log_part_offset.
CurrentMemory	Indicates the amount of the server memory currently in use.
DBFileName	Contains the database filename.
DBImplementationClass	Indicates the database implementation class number.
DBImplementationNo	Indicates the database implementation number.
DBSiteName	Contains the database site name.
DeleteCount	Holds the number of database deletes since the database was last attached.
ExpungeCount	Indicates the number of removals of a record and all of its ancestors.
Fetches	Indicates the number of reads from the memory buffer cache.
ForcedWrites	Used to indicate the mode in which database writes are performed.
InsertCount	Holds the number of insertions into the database since the database was last attached.
IsRemoteConnect	Used to indicate whether the connection with the database is remote.
LogFile	Used to indicate the value of isc_info_log_file.
Marks	Used to indicate the number of writes to the memory buffer cache.
MaxMemory	Indicates the maximum amount of memory used at one time since the first process attached to the database.

NoReserve	Specifies if the space for holding backup versions of modified records is reserved on each database page.
NumBuffers	Indicates the number of currently allocated memory buffers.
NumWALBuffers	Indicates the value of isc info num wal buffers.
ODSMajorVersion	Indicates the on disk structure (ODS) major version number.
ODSMinorVersion	Indicates the on disk structure (ODS) minor version number.
PageSize	Shows the number of bytes per page for the database.
PurgeCount	Holds the number of removals of records committed from the database, resulting in older versions.
ReadIdxCount	Holds the number of reads done via an index since the database was last attached.
ReadOnly	Indicates whether the database is read only.
Reads	Indicates the number of page reads from the database since the current database was first attached.
ReadSeqCount	Contains the number of sequential database reads done on each table since the database was last attached.
SweepInterval	Indicates the number of transactions that are committed between "sweeps".
UpdateCount	Contains the number of database updates since the database was last attached.
UserNames	Contains the names of all

	users currently attached to the database.
Version	Indicates the version of the database implementation.
WALAverageGroupCommitSize	Indicates the value of <code>isc_info_wal_avg_grpc_size</code> .
WALAverageIOSize	Indicates the value of <code>isc_info_wal_avg_io_size</code> .
WALBufferSize	Indicates the value of <code>isc_info_wal_buffer_size</code> .
WALCheckpointLength	Indicates the value of <code>isc_info_wal_ckpt_length</code> .
WALCurCheckpointInterval	Indicates the value of <code>isc_info_wal_cur_ckpt_interval</code> .
WALGroupCommitWaitUSecs	Indicates the value of <code>isc_info_wal_grpc_wait_usecs</code> .
WALNumCommits	Indicates the value of <code>isc_info_wal_num_commits</code> .
WALNumIO	Indicates the value of <code>isc_info_wal_num_id</code> .
WALPrvCheckpointFilename	Indicates the value of <code>isc_info_wal_prv_ckpt_filename</code> .
WALPrvCheckpointPartOffset	Indicates the value of <code>isc_info_wal_prv_ckpt_offset</code> .
Writes	Indicates the number of page writes to the current database since it was first attached by any process.

See Also

- [TGDSDatabaseInfo Class](#)
- [TGDSDatabaseInfo Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.1 Allocation Property

Indicates the number of database pages allocated.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property Allocation: integer;
```

Remarks

Use the Allocation property to determine the number of database pages allocated.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.2 AttachmentID Property

Indicates a unique connection identifier.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property AttachmentID: integer;
```

Remarks

Use the AttachmentID property to indicate a unique connection identifier.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.3 BackoutCount Property

Contains the number of removals of a record version.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property BackoutCount: TStringList;
```

Remarks

Use the BackoutCount property to indicate the number of removals of a record version.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.4 BaseLevel Property

Indicates the database version number.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property BaseLevel: integer;
```

Remarks

Use the BaseLevel property to determine the database version number.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.5 CurLogFileName Property

Contains the name of the log-file.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property CurLogFileName: string;
```

Remarks

Contains the name of the log-file. The TIBCCConnection component must be active, otherwise

an exception is raised.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.6 CurLogPartitionOffset Property

Indicates the value of `isc_info_cur_log_part_offset`.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property CurLogPartitionOffset: integer;
```

Remarks

Use the `CurLogPartitionOffset` property to determine the value of `isc_info_cur_log_part_offset`.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.7 CurrentMemory Property

Indicates the amount of the server memory currently in use.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property CurrentMemory: integer;
```

Remarks

Use the `CurrentMemory` property to determine the amount of server memory (in bytes) currently in use.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.8 DBFileName Property

Contains the database filename.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property DBFileName: string;
```

Remarks

Contains the database filename.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.9 DBImplementationClass Property

Indicates the database implementation class number.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property DBImplementationClass: integer;
```

Remarks

Use the DBImplementationClass property to determine the database implementation class number.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.10 DBImplementationNo Property

Indicates the database implementation number.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property DBImplementationNo: integer;
```

Remarks

Use the DBImplementationNo to determine the database implementation number.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.11 DBSiteName Property

Contains the database site name.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property DBSiteName: string;
```

Remarks

Contains the database site name.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.12 DeleteCount Property

Holds the number of database deletes since the database was last attached.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property DeleteCount: TStringList;
```

Remarks

Contains the number of database deletes since the database was last attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.13 ExpungeCount Property

Indicates the number of removals of a record and all of its ancestors.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property ExpungeCount: TStringList;
```

Remarks

Use the ExpungeCount property to determine the number of removals of a record and all of its ancestors.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.14 Fetches Property

Indicates the number of reads from the memory buffer cache.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property Fetches: integer;
```

Remarks

Use the Fetches property to determine the number of reads from the memory buffer cache.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.15 ForcedWrites Property

Used to indicate the mode in which database writes are performed.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property Forcedwrites: integer;
```

Remarks

Use the FoprcedWrites property to indicate the mode in which database writes are performed. It is 0 for asynchronous mode, 1 for synchronous mode

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.16 InsertCount Property

Holds the number of insertions into the database since the database was last attached.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property InsertCount: TStringList;
```

Remarks

Contains the number of insertions into the database since the database was last attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.17 IsRemoteConnect Property

Used to indicate whether the connection with the database is remote.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property IsRemoteConnect: Boolean;
```

Remarks

Use the IsRemoteConnect property to determine whether the connection with the database is remote.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.18 LogFile Property

Used to indicate the value of isc_info_log_file.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property LogFile: integer;
```

Remarks

Use the LogFile property to indicate the value of isc_info_log_file.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.19 Marks Property

Used to indicate the number of writes to the memory buffer cache.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property Marks: integer;
```

Remarks

Use the Marks property to determine the number of writes to the memory buffer cache.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.20 MaxMemory Property

Indicates the maximum amount of memory used at one time since the first process attached to the database.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property MaxMemory: integer;
```

Remarks

Use the MaxMemory property to indicate the maximum amount of memory used at one time since the first process attached to the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.21 NoReserve Property

Specifies if the space for holding backup versions of modified records is reserved on each database page.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property NoReserve: integer;
```

Remarks

If 0, the space is reserved on each database page for holding backup versions of modified

records.

If 1, no space is reserved for such records.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.22 NumBuffers Property

Indicates the number of currently allocated memory buffers.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property NumBuffers: integer;
```

Remarks

Use the NumBuffers property to indicate the number of currently allocated memory buffers.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.23 NumWALBuffers Property

Indicates the value of isc_info_num_wal_buffers.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property NumWALBuffers: integer;
```

Remarks

Use the NumWALBuffers property to indicate the value of isc_info_num_wal_buffers.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.24 ODSMajorVersion Property

Indicates the on disk structure (ODS) major version number.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property ODSMajorVersion: integer;
```

Remarks

Use the ODSMajorVersion to determine the on disk structure (ODS) major version number.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.25 ODSMinorVersion Property

Indicates the on disk structure (ODS) minor version number.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property ODSMinorVersion: integer;
```

Remarks

Use the ODSMinorVersion property to determine the on disk structure (ODS) minor version number.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.26 PageSize Property

Shows the number of bytes per page for the database.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property PageSize: integer;
```

Remarks

Shows the number of bytes per page for the database.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.27 PurgeCount Property

Holds the number of removals of records committed from the database, resulting in older versions.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property PurgeCount: TStringList;
```

Remarks

Contains the number of removals of records committed from the database, resulting in older versions.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.28 ReadIdxCount Property

Holds the number of reads done via an index since the database was last attached.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property ReadIdxCount: TStringList;
```

Remarks

The ReadIdxCount property is used to contain the number of reads done via an index since the database was last attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.29 ReadOnly Property

Indicates whether the database is read only.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property ReadOnly: Boolean;
```

Remarks

The ReadOnly property is used to indicate whether the database is read only.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.30 Reads Property

Indicates the number of page reads from the database since the current database was first attached.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property Reads: integer;
```

Remarks

The Reads property is used to indicate the number of page reads from the database since the current database was first attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.31 ReadSeqCount Property

Contains the number of sequential database reads done on each table since the database was last attached.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property ReadSeqCount: TStringList;
```

Remarks

The ReadSeqCount property is used to contain the number of sequential database reads done on each table since the database was last attached.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.32 SweepInterval Property

Indicates the number of transactions that are committed between "sweeps".

Class

[TGDSDatabaseInfo](#)

Syntax

```
property SweepInterval: integer;
```

Remarks

Use the SweepInterval property to indicate the number of transactions that are committed between "sweeps".

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.33 UpdateCount Property

Contains the number of database updates since the database was last attached.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property UpdateCount: TStringList;
```

Remarks

The UpdateCount property is used to hold the number of database updates since the database was last attached.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.34 UserNames Property

Contains the names of all users currently attached to the database.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property UserNames: TStringList;
```

Remarks

The UserNames property is used to hold the names of all users currently attached to the database.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.35 Version Property

Indicates the version of the database implementation.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property Version: string;
```

Remarks

The Version property is used to indicate the version of the database implementation.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.36 WALAverageGroupCommitSize Property

Indicates the value of isc_info_wal_avg_grpc_size.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property WALAverageGroupCommitSize: integer;
```

Remarks

The WALAverageGroupCommitSize property is used to indicate the value of
isc_info_wal_avg_grpc_size.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.37 WALAverageIOSize Property

Indicates the value of isc_info_wal_avg_io_size.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property WALAverageIOSize: integer;
```

Remarks

The WALAverageIOSize property is used to indicate the value of isc_info_wal_avg_io_size.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.38 WALBufferSize Property

Indicates the value of isc_info_wal_buffer_size.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property WALBufferSize: integer;
```

Remarks

The WALBufferSize property is used to indicate the value of isc_info_wal_buffer_size.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.39 WALCheckpointLength Property

Indicates the value of isc_info_wal_ckpt_length.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property WALCheckpointLength: integer;
```

Remarks

The WALCheckpointLength is used to indicate the value of isc_info_wal_ckpt_length.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.40 WALCurCheckpointInterval Property

Indicates the value of isc_info_wal_cur_ckpt_interval.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property WALCurCheckpointInterval: integer;
```

Remarks

The WALCurCheckpointInterval property is used to indicate the value of isc_info_wal_cur_ckpt_interval.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.41 WALGroupCommitWaitUsecs Property

Indicates the value of isc_info_wal_grpc_wait_usecs.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property WALGroupCommitwaitUsecs: integer;
```

Remarks

The WALGroupCommitWaitUsec property is used to indicate the value of isc_info_wal_grpc_wait_usecs.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.42 WALNumCommits Property

Indicates the value of `isc_info_wal_num_commits`.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property WALNumCommits: integer;
```

Remarks

The WALNumCommits property is used to indicate the value of `isc_info_wal_num_commits`.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.43 WALNumIO Property

Indicates the value of `isc_info_wal_num_id`.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property WALNumIO: integer;
```

Remarks

The WALNumI property is used to indicate the value of `isc_info_wal_num_id`.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.44 WALPrvCheckpointFilename Property

Indicates the value of `isc_info_wal_prv_ckpt_fname`.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property WALPrvCheckpointFilename: string;
```

Remarks

The WALPrvCheckpointFilename property is used to indicate the value of isc_info_wal_prv_ckpt_fname.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.45 WALPrvCheckpointPartOffset Property

Indicates the value of isc_info_wal_prv_ckpt_poffset.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property WALPrvCheckpointPartOffset: integer;
```

Remarks

The WALPrvCheckpointPartOffset property is used to indicate the value of isc_info_wal_prv_ckpt_poffset.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.1.2.46 Writes Property

Indicates the number of page writes to the current database since it was first attached by any process.

Class

[TGDSDatabaseInfo](#)

Syntax

```
property writes: integer;
```

Remarks

The Writes property is used to indicate the number of page writes to the current database since it was first attached by any process.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2 TIBCBlob Class

A class holding value of the BLOB fields and parameters.

For a list of all members of this type, see [TIBCBlob](#) members.

Unit

[IBCClasses](#)

Syntax

```
TIBCBlob = class (TCompressedBlob);
```

Remarks

TIBCBlob is a descendant of TCompressedBlob class. It holds value of the BLOB fields and parameters.

Inheritance Hierarchy

[TSharedObject](#)

[TBlob](#)

[TCompressedBlob](#)

TIBCBlob

See Also

- [TCompressedBlob](#)
- [TCustomIBCDataSet.GetBlob](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.1 Members

[TIBCBlob](#) class overview.

Properties

Name	Description
AsString (inherited from TBlob)	Used to manipulate BLOB value as string.
AsWideString (inherited from TBlob)	Used to manipulate BLOB value as Unicode string.
Cached	Indicates whether the BLOB data are cached on the client or they are accessed remotely on the server.
CharsetID	Source charset for BLOB subtype conversion using BLOB filters.
Compressed (inherited from TCompressedBlob)	Used to indicate if the Blob is compressed.
CompressedSize (inherited from TCompressedBlob)	Used to indicate compressed size of the Blob data.
ConversionCharsetID	Target charset for BLOB subtype conversion using Blob filters.
ConversionSubType	Target subtype for the BLOB subtype conversion using BLOB filters.
DbHandle	Contains the handle of the database where the BLOB is stored.
Handle	Contains the BLOB handle.
ID	Contains BLOB ID.
IsUnicode (inherited from TBlob)	Gives choice of making TBlob store and process data in Unicode format or not.
MaxSegmentSize	Indicates the largest BLOB segment size.
NumSegments	Indicates the number of the BLOB segments in the database.

RefCount (inherited from TSharedObject)	Used to return the count of reference to a TSharedObject object.
Size (inherited from TBlob)	Used to learn the size of the TBlob value in bytes.
Streamed	Indicates whether the BLOB is stream or segmented.
SubType	Source subtype for the BLOB subtype conversion using BLOB filters.
TrHandle	Contains the handle of the transaction in which the BLOB is read or written.

Methods

Name	Description
AddRef (inherited from TSharedObject)	Increments the reference count for the number of references dependent on the TSharedObject object.
AllocBlob	Allocates and initializes the Blob handle.
Assign (inherited from TBlob)	Sets BLOB value from another TBlob object.
Clear (inherited from TBlob)	Deletes the current value in TBlob object.
CloseBlob	Frees the Blob handle.
FreeBlob	Clears BLOB ID.
IsInit	Verifies BLOB ID initialization.
LengthBlob	Determines the number of bytes contained in the Blob object.
LoadFromFile (inherited from TBlob)	Loads the contents of a file into a TBlob object.
LoadFromStream (inherited from TBlob)	Copies the contents of a stream into the TBlob object.
Read (inherited from TBlob)	Acquires a raw sequence of bytes from the data stored in TBlob.
ReadBlob	Reads BLOB from the

	database.
Release (inherited from TSharedObject)	Decrements the reference count.
SaveToFile (inherited from TBlob)	Saves the contents of the TBlob object to a file.
SaveToStream (inherited from TBlob)	Copies the contents of a TBlob object to a stream.
Truncate (inherited from TBlob)	Sets new TBlob size and discards all data over it.
Write (inherited from TBlob)	Stores a raw sequence of bytes into a TBlob object.
WriteBlob	Writes BLOB to the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.15.1.2.2 Properties

Properties of the **TIBCBlob** class.

For a complete list of the **TIBCBlob** class members, see the [TIBCBlob Members](#) topic.

Public

Name	Description
AsString (inherited from TBlob)	Used to manipulate BLOB value as string.
AsWideString (inherited from TBlob)	Used to manipulate BLOB value as Unicode string.
Cached	Indicates whether the BLOB data are cached on the client or they are accessed remotely on the server.
CharsetID	Source charset for BLOB subtype conversion using BLOB filters.
Compressed (inherited from TCompressedBlob)	Used to indicate if the Blob is compressed.
CompressedSize (inherited from TCompressedBlob)	Used to indicate compressed size of the Blob data.
ConversionCharsetID	Target charset for BLOB

	subtype conversion using Blob filters.
ConversionSubType	Target subtype for the BLOB subtype conversion using BLOB filters.
DbHandle	Contains the handle of the database where the BLOB is stored.
Handle	Contains the BLOB handle.
ID	Contains BLOB ID.
IsUnicode (inherited from TBlob)	Gives choice of making TBlob store and process data in Unicode format or not.
MaxSegmentSize	Indicates the largest BLOB segment size.
NumSegments	Indicates the number of the BLOB segments in the database.
RefCount (inherited from TSharedObject)	Used to return the count of reference to a TSharedObject object.
Size (inherited from TBlob)	Used to learn the size of the TBlob value in bytes.
Streamed	Indicates whether the BLOB is stream or segmented.
SubType	Source subtype for the BLOB subtype conversion using BLOB filters.
TrHandle	Contains the handle of the transaction in which the BLOB is read or written.

See Also

- [TIBCBlob Class](#)
- [TIBCBlob Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.2.1 Cached Property

Indicates whether the BLOB data are cached on the client or they are accessed remotely on the server.

Class

[TIBCBlob](#)

Syntax

```
property Cached: Boolean;
```

Remarks

The Cached property is used to indicate whether the BLOB data are cached on the client or they are accessed remotely on the server. In most cases you don't need to set the value of this property directly. To enable or disable BLOB caching, use the [TCustomIBDataSet.Options](#) property.

See Also

- [TCustomIBDataSet.Options](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.2.2 CharSetID Property

Source charset for BLOB subtype conversion using BLOB filters.

Class

[TIBCBlob](#)

Syntax

```
property CharSetID: integer;
```

Remarks

Source charset for BLOB subtype conversion using BLOB filters. It corresponds to `isc_bpb_source_interp` parameter in BLOB Parameter Buffer (BPB) that is used for filtration. For more information on BLOB filters refer to InterBase API Guide.

See Also

- [ConversionCharsetID](#)
- [SubType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.2.3 ConversionCharsetID Property

Target charset for BLOB subtype conversion using Blob filters.

Class

[TIBCBlob](#)

Syntax

```
property ConversionCharsetID: integer;
```

Remarks

Target charset for BLOB subtype conversion using Blob filters. It corresponds to `isc_bpb_target_interp` parameter in BLOB Parameter Buffer (BPB) that is used for filtration. For more information on BLOB filters refer to InterBase API Guide.

Set ConversionCharset property to the charset that you want to get when reading the BLOB from database.

Set ConversionCharset property to the charset that you want BLOB to store to database in when writing BLOB to database.

See Also

- [CharsetID](#)
- [ConversionSubType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.2.4 ConversionSubType Property

Target subtype for the BLOB subtype conversion using BLOB filters.

Class

[TIBCBlob](#)

Syntax

```
property ConversionSubType: integer;
```

Remarks

Target subtype for the BLOB subtype conversion using BLOB filters. It corresponds to the *isc_bpb_target_type* parameter in BLOB Parameter Buffer (BPB) that is used for filtration. For more information about Blob subtypes and filters refer to InterBase API Guide.

Set the ConversionSubType property to the Blob subtype that you want to get when reading the Blob from database.

Set the ConversionSubType property to the Blob subtype that you want to store to database when writing Blob to database.

See Also

- [SubType](#)
- [ConversionCharsetID](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.2.5 DbHandle Property

Contains the handle of the database where the BLOB is stored.

Class

[TIBCBlob](#)

Syntax

```
property DbHandle: TISC_DB_HANDLE;
```

Remarks

The DbHandle property is used to contain the handle of the database where the BLOB is stored.

See Also

- [TIBCConnection.Handle](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.2.6 Handle Property

Contains the BLOB handle.

Class

[TIBCBlob](#)

Syntax

```
property Handle: TISC_BLOB_HANDLE;
```

Remarks

Contains the BLOB handle. Use the Handle property for direct calls to InterBase BLOB API.

See Also

- [AllocBlob](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.2.7 ID Property

Contains BLOB ID.

Class

[TIBCBlob](#)

Syntax

```
property ID: TISC_QUAD;
```

Remarks

Contains BLOB ID that is actually stored in the BLOB field of the table record. It is a unique numeric value that references the BLOB data.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.2.8 MaxSegmentSize Property

Indicates the largest BLOB segment size.

Class

[TIBCBlob](#)

Syntax

```
property MaxSegmentSize: word;
```

Remarks

The MaxSegmentSize property is used to indicate the size in bytes of the largest segment of the BLOB.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.2.9 NumSegments Property

Indicates the number of the BLOB segments in the database.

Class

[TIBCBlob](#)

Syntax

```
property NumSegments: Cardinal;
```

Remarks

Use the NumSegments property to indicate the number of the BLOB segments in the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.2.10 Streamed Property

Indicates whether the BLOB is stream or segmented.

Class

[TIBCBlob](#)

Syntax

```
property Streamed: Boolean;
```

Remarks

Use the Streamed property to determine whether the BLOB is stream or segmented. Segmented BLOBs are usual InterBase BLOBs and are stored in chunks. Stream BLOBs are stored as a continuous array of data bytes.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.2.11 SubType Property

Source subtype for the BLOB subtype conversion using BLOB filters.

Class

[TIBCBlob](#)

Syntax

```
property SubType: integer;
```

Remarks

Source subtype for the BLOB subtype conversion using BLOB filters. It corresponds to *isc_bpb_source_type* parameter in BLOB Parameter Buffer (BPB) that is used for filtration.

For more information on BLOB subtypes and filters refer to InterBase API Guide.

See Also

- [ConversionSubType](#)
- [CharsetID](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.2.12 TrHandle Property

Contains the handle of the transaction in which the BLOB is read or written.

Class

[TIBCBlob](#)

Syntax

```
property TrHandle: TISC_TR_HANDLE;
```

Remarks

Use the TrHandle property to hold the handle of the transaction in which the BLOB is read or written.

See Also

- [TIBCTransaction.Handle](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.3 Methods

Methods of the **TIBCBlob** class.

For a complete list of the **TIBCBlob** class members, see the [TIBCBlob Members](#) topic.

Public

Name	Description
------	-------------

AddRef (inherited from TSharedObject)	Increments the reference count for the number of references dependent on the TSharedObject object.
AllocBlob	Allocates and initializes the Blob handle.
Assign (inherited from TBlob)	Sets BLOB value from another TBlob object.
Clear (inherited from TBlob)	Deletes the current value in TBlob object.
CloseBlob	Frees the Blob handle.
FreeBlob	Clears BLOB ID.
IsInit	Verifies BLOB ID initialization.
LengthBlob	Determines the number of bytes contained in the Blob object.
LoadFromFile (inherited from TBlob)	Loads the contents of a file into a TBlob object.
LoadFromStream (inherited from TBlob)	Copies the contents of a stream into the TBlob object.
Read (inherited from TBlob)	Acquires a raw sequence of bytes from the data stored in TBlob.
ReadBlob	Reads BLOB from the database.
Release (inherited from TSharedObject)	Decrements the reference count.
SaveToFile (inherited from TBlob)	Saves the contents of the TBlob object to a file.
SaveToStream (inherited from TBlob)	Copies the contents of a TBlob object to a stream.
Truncate (inherited from TBlob)	Sets new TBlob size and discards all data over it.
Write (inherited from TBlob)	Stores a raw sequence of bytes into a TBlob object.
WriteBlob	Writes BLOB to the database.

See Also

- [TIBCBlob Class](#)

- [TIBCBlob Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.3.1 AllocBlob Method

Allocates and initializes the Blob handle.

Class

[TIBCBlob](#)

Syntax

```
procedure AllocBlob(Read: Boolean = True);
```

Parameters

Read

True, if BLOB is opened for reading or writing.

Remarks

Call the AllocBLOB method to allocate and initialize the Blob handle. The Read parameter indicates whether the BLOB is opened for reading or writing. If the BLOB is opened for writing, new BLOB ID is generated.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.3.2 CloseBlob Method

Frees the Blob handle.

Class

[TIBCBlob](#)

Syntax

```
procedure CloseBlob;
```

Remarks

Use the CloseBLOB method to free the Blob handle.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.3.3 FreeBlob Method

Clears BLOB ID.

Class

[TIBCBlob](#)

Syntax

```
procedure FreeBlob; override;
```

Remarks

Call the FreeBLOB method to clear BLOB ID.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.3.4 IsInit Method

Verifies BLOB ID initialization.

Class

[TIBCBlob](#)

Syntax

```
function IsInit: Boolean;
```

Return Value

True, if BLOB ID is initialized. False otherwise.

Remarks

The IsInit method verifies that BLOB ID is initialized.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.3.5 LengthBlob Method

Determines the number of bytes contained in the Blob object.

Class

[TIBCBlob](#)

Syntax

```
function LengthBlob: integer;
```

Return Value

The number of bytes contained in the Blob object.

Remarks

The LengthBLOB method returns the number of bytes contained in the Blob object.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.3.6 ReadBlob Method

Reads BLOB from the database.

Class

[TIBCBlob](#)

Syntax

```
procedure ReadBlob;
```

Remarks

Call the ReadBLOB method to read BLOB from the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.1.2.3.7 WriteBlob Method

Writes BLOB to the database.

Class

[TIBCBlob](#)

Syntax

```
procedure writeBlob;
```

Remarks

Call the WriteBLOB method to write BLOB to the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.2 Enumerations

Enumerations in the **IBCClasses** unit.

Enumerations

Name	Description
TIBCIsoationLevel	Specifies the transaction isolation level and access mode.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.2.1 TIBCIsoationLevel Enumeration

Specifies the transaction isolation level and access mode.

Unit

[IBCClasses](#)

Syntax

```
TIBCIsoationLevel = (ibISnapshot, ibIReadCommitted,
```

```
iblReadOnlyReadCommitted, iblTableStability,
iblReadOnlyTableStability, iblCustom);
```

Values

Value	Meaning
iblCustom	The parameters of the transaction are set manually in the Params property.
iblReadCommitted	Enables the transaction to see all committed data in the database, and to update rows updated and committed by other simultaneous transactions without causing lost update problems.
iblReadOnlyReadCommitted	Enables the transaction to see all committed data in the database with read-only access mode.
iblReadOnlyTableStability	Provides a transaction read-only access to the tables it uses. Other simultaneous transactions may be able to select rows from these tables, but they can not insert, update, and delete rows from these tables.
iblSnapshot	The default isolation level. Provides a stable, committed view of the database at the time the transaction starts. Other simultaneous transactions can UPDATE and INSERT rows, but this transaction cannot see these changes. For updated rows, this transaction sees versions of these rows as they existed at the start of the transaction. If this transaction attempts to update or delete rows changed by another transaction, an update conflict is reported.
iblTableStability	Provides a transaction sole insert, update, and delete access to the tables it uses. Other simultaneous transactions may still be able to select rows from these tables.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.3 Routines

Routines in the **IBCClasses** unit.

Routines

Name	Description
Reverse2	Switches places of bytes in the word argument.
Reverse4	Switches places of bytes in the cardinal argument.

[XSQLDA_LENGTH](#)

Analogue of InterBase XSQLDA_LENGTH macro. Calculates the number of bytes that must be allocated for an input or output XSQLDA.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.3.1 Reverse2 Procedure

Switches places of bytes in the word argument.

Unit

[IBCClasses](#)

Syntax

Parameters

Value

Holds the input value.

Return Value

the output value.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.3.2 Reverse4 Function

Switches places of bytes in the cardinal argument.

Unit

[IBCClasses](#)

Syntax

```
function Reverse4(Value: cardinal): cardinal;
```

Parameters

Value

Holds the input value.

Return Value

the output value.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.3.3 XSQLDA_LENGTH Function

Analogue of InterBase XSQLDA_LENGTH macro. Calculates the number of bytes that must be allocated for an input or output XSQLDA.

Unit

[IBCClasses](#)

Syntax

```
function XSQLDA_LENGTH(n: Long; XSQLVARType: TXSQLVARType): Long;
```

Parameters

n

Holds the count of XSQLVAR.

XSQLVARType

Holds the type of XSQLVAR.

Return Value

the number of bytes that must be allocated for an input or output XSQLDA.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.15.4 Variables

Variables in the **IBCClasses** unit.

Variables

Name	Description
IntegerPrecision	Set this constant to define the type of NUMERIC and DECIMAL fields with precision less or equal than IntegerPrecision as dtInteger. Otherwise, they are defined as dtFloat.

© 1997-2024
 Devart. All Rights Reserved. [Request Support](#) [DAC Forum](#) [Provide Feedback](#)

5.15.4.1 IntegerPrecision Variable

Set this constant to define the type of NUMERIC and DECIMAL fields with precision less or equal than IntegerPrecision as dtInteger. Otherwise, they are defined as dtFloat.

Unit

[IBCClasses](#)

Syntax

```
IntegerPrecision: integer = 10;
```

© 1997-2024
 Devart. All Rights Reserved. [Request Support](#) [DAC Forum](#) [Provide Feedback](#)

5.16 IBConnectionPool

This unit contains the TIBConnectionPoolManager class for managing connection pool.

Classes

Name	Description
TIBConnectionPoolManager	A class of methods that are used for managing IBDAC connection pool.

© 1997-2024
 Devart. All Rights Reserved. [Request Support](#) [DAC Forum](#) [Provide Feedback](#)

5.16.1 Classes

Classes in the **IBConnectionPool** unit.

Classes

Name	Description
TIBConnectionPoolManager	A class of methods that are used for managing IBDAC

	connection pool.
--	------------------

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.16.1.1 TIBConnectionPoolManager Class

A class of methods that are used for managing IBDAC connection pool.

For a list of all members of this type, see [TIBConnectionPoolManager](#) members.

Unit

[IBConnectionPool](#)

Syntax

```
TIBConnectionPoolManager = class(TCRConnectionPoolManager);
```

Remarks

Use the TIBConnectionPoolManager methods to manage IBDAC connection pool.

Inheritance Hierarchy

TCRConnectionPoolManager

TIBConnectionPoolManager

See Also

- [Connection Pooling](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.16.1.1.1 Members

[TIBConnectionPoolManager](#) class overview.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.17 IBCDataTypeMap

This unit contains the implementation of mapping between InterBase/Firebird and Delphi data types.

Constants

Name	Description
IBCArray	This unit contains the TIBCArray class for working the InterBase/Firebird array data types.
ibcArray	Used to map ARRAY to Delphi data types.
ibcBigint	Used to map BIGINT to Delphi data types.
ibcBlob	Used to map BLOB to Delphi data types.
ibcBoolean	Used to map BOOLEAN to Delphi data types.
ibcChar	Used to map CHAR to Delphi data types.
ibcCharBin	Used to map BINARY to Delphi data types.
ibcDate	Used to map DATE to Delphi data types.
ibcDecimal	Used to map DECIMAL to Delphi data types.
ibcDouble	Used to map DOUBLE PRECISION to Delphi data types.
ibcFloat	Used to map FLOAT to Delphi data types.
ibcInteger	Used to map INTEGER to Delphi data types.
ibcNumeric	Used to map NUMERIC to Delphi data types.
ibcSmallint	Used to map SMALLINT to Delphi data types.
ibcText	Used to map TEXT data types to Delphi data types.
ibcTime	Used to map TIME to Delphi data types.

ibcTimestamp	Used to map TIMESTAMP to Delphi data types.
ibcVarchar	Used to map VARCHAR to Delphi data types.
ibcVarcharBin	Used to map VARCHAR BINARY to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1 Constants

Constants in the **IBCDataTypeMap** unit.

Constants

Name	Description
IBCArray	This unit contains the TIBCArray class for working the InterBase/Firebird array data types.
ibcArray	Used to map ARRAY to Delphi data types.
ibcBigint	Used to map BIGINT to Delphi data types.
ibcBlob	Used to map BLOB to Delphi data types.
ibcBoolean	Used to map BOOLEAN to Delphi data types.
ibcChar	Used to map CHAR to Delphi data types.
ibcCharBin	Used to map BINARY to Delphi data types.
ibcDate	Used to map DATE to Delphi data types.
ibcDecimal	Used to map DECIMAL to Delphi data types.
ibcDouble	Used to map DOUBLE PRECISION to Delphi data types.
ibcFloat	Used to map FLOAT to

	Delphi data types.
ibcInteger	Used to map INTEGER to Delphi data types.
ibcNumeric	Used to map NUMERIC to Delphi data types.
ibcSmallint	Used to map SMALLINT to Delphi data types.
ibcText	Used to map TEXT data types to Delphi data types.
ibcTime	Used to map TIME to Delphi data types.
ibcTimestamp	Used to map TIMESTAMP to Delphi data types.
ibcVarchar	Used to map VARCHAR to Delphi data types.
ibcVarcharBin	Used to map VARCHAR BINARY to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.1 IBCArray Constant

This unit contains the TIBCArray class for working the InterBase/Firebird array data types.

Classes

Name	Description
TCustomIBCArray	A base class representing the value of the InterBase array data type.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.2 ibcArray Constant

Used to map **ARRAY** to Delphi data types.

Unit

[IBCDataTypeMap](#)

Syntax

```
ibcArray = ibcBase + 18;
```

Remarks

Use the **ibcArray** constant to map the InterBase/Firebird **ARRAY** data types to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.3 **ibcBigint** Constant

Used to map **BIGINT** to Delphi data types.

Unit

[IBCDatatypeMap](#)

Syntax

```
ibcBigint = ibcBase + 4;
```

Remarks

Use the **ibcBigint** constant to map the Firebird **BIGINT** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.4 **ibcBlob** Constant

Used to map **BLOB** to Delphi data types.

Unit

[IBCDatatypeMap](#)

Syntax

```
ibcBlob = ibcBase + 16;
```

Remarks

Use the **ibcBlob** constant to map the InterBase/Firebird **BLOB** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.5 **ibcBoolean Constant**

Used to map **BOOLEAN** to Delphi data types.

Unit

[IBCDatatypeMap](#)

Syntax

```
ibcBoolean = ibcBase + 1;
```

Remarks

Use the **ibcBoolean** constant to map the InterBase/Firebird **BOOLEAN** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.6 **ibcChar Constant**

Used to map **CHAR** to Delphi data types.

Unit

[IBCDatatypeMap](#)

Syntax

```
ibcChar = ibcBase + 12;
```

Remarks

Use the **ibcChar** constant to map the InterBase/Firebird **CHAR** data type to Delphi data types.

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.17.1.7 **ibcCharBin** Constant

Used to map **BINARY** to Delphi data types.

Unit

[IBCDDataTypeMap](#)

Syntax

```
ibcCharBin = ibcBase + 14;
```

Remarks

Use the **ibcCharBin** constant to map the InterBase/Firebird **BINARY** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.8 **ibcDate** Constant

Used to map **DATE** to Delphi data types.

Unit

[IBCDDataTypeMap](#)

Syntax

```
ibcDate = ibcBase + 9;
```

Remarks

Use the **ibcDate** constant to map the InterBase/Firebird **DATE** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.9 **ibcDecimal** Constant

Used to map **DECIMAL** to Delphi data types.

Unit

[IBCDatatypeMap](#)

Syntax

```
ibcDecimal = ibcBase + 7;
```

Remarks

Use the **ibcDecimal** constant to map the InterBase/Firebird **DECIMAL** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.10 **ibcDouble** Constant

Used to map **DOUBLE PRECISION** to Delphi data types.

Unit

[IBCDatatypeMap](#)

Syntax

```
ibcDouble = ibcBase + 6;
```

Remarks

Use the **ibcDouble** constant to map the InterBase/Firebird **DOUBLE PRECISION** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.11 ibcFloat Constant

Used to map **FLOAT** to Delphi data types.

Unit

[IBCDataTypeMap](#)

Syntax

```
ibcFloat = ibcBase + 5;
```

Remarks

Use the **ibcFloat** constant to map the InterBase/Firebird **FLOAT** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.12 ibcInteger Constant

Used to map **INTEGER** to Delphi data types.

Unit

[IBCDataTypeMap](#)

Syntax

```
ibcInteger = ibcBase + 3;
```

Remarks

Use the **ibcInteger** constant to map the InterBase/Firebird **INTEGER** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.13 **ibcNumeric Constant**

Used to map **NUMERIC** to Delphi data types.

Unit

[IBCDatatypeMap](#)

Syntax

```
ibcNumeric = ibcBase + 8;
```

Remarks

Use the **ibcNumeric** constant to map the InterBase/Firebird **NUMERIC** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.14 **ibcSmallint Constant**

Used to map **SMALLINT** to Delphi data types.

Unit

[IBCDatatypeMap](#)

Syntax

```
ibcSmallint = ibcBase + 2;
```

Remarks

Use the **ibcSmallint** constant to map the InterBase/Firebird **SMALLINT** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.15 **ibcText** Constant

Used to map **TEXT** data types to Delphi data types.

Unit

[IBCDatatypeMap](#)

Syntax

```
ibcText = ibcBase + 17;
```

Remarks

Use the **ibcText** constant to map the InterBase/Firebird **TEXT** data types to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.16 **ibcTime** Constant

Used to map **TIME** to Delphi data types.

Unit

[IBCDatatypeMap](#)

Syntax

```
ibcTime = ibcBase + 10;
```

Remarks

Use the **ibcTime** constant to map the InterBase/Firebird **TIME** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.17 **ibcTimestamp** Constant

Used to map **TIMESTAMP** to Delphi data types.

Unit

[IBCDataTypeMap](#)

Syntax

```
ibcTimestamp = ibcBase + 11;
```

Remarks

Use the **ibcTimestamp** constant to map the InterBase/Firebird **TIMESTAMP** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.18 **ibcVarchar Constant**

Used to map **VARCHAR** to Delphi data types.

Unit

[IBCDataTypeMap](#)

Syntax

```
ibcVarchar = ibcBase + 13;
```

Remarks

Use the **ibcVarchar** constant to map the InterBase/Firebird **VARCHAR** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.17.1.19 **ibcVarcharBin Constant**

Used to map **VARCHAR BINARY** to Delphi data types.

Unit

[IBCDataTypeMap](#)

Syntax

```
ibcVarcharBin = ibcBase + 15;
```

Remarks

Use the **ibcVarcharBin** constant to map the InterBase/Firebird **VARCHAR BINARY** data type to Delphi data types.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.18 IBCErrors

IBCErrors unit implements the class.

Classes

Name	Description
EIBCErrors	A base class for exceptions raised when a component detects an InterBase error.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.18.1 Classes

Classes in the **IBCErrors** unit.

Classes

Name	Description
EIBCErrors	A base class for exceptions raised when a component detects an InterBase error.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.18.1.1 EIBCErrors Class

A base class for exceptions raised when a component detects an InterBase error.

For a list of all members of this type, see [EIBCErrors](#) members.

Unit

[IBCErrors](#)

Syntax

```
EIBCErrors = class(EDAError);
```

Remarks

EIBCErrors is raised when a component detects an InterBase error. Use EIBCErrors in an exception handling block.

Inheritance Hierarchy

[EDAError](#)

EIBCErrors

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.18.1.1.1 Members

[EIBCErrors](#) class overview.

Properties

Name	Description
Component (inherited from EDAError)	Contains the component that caused the error.
ErrorCode (inherited from EDAError)	Determines the error code returned by the server.
ErrorNumber	Used to determine the error number returned by InterBase.
Sender	Holds the reference to the sender if exception is raised by the TComponent instance.
SQLErrorMsg	Holds the error message describing the part of the SQL code that caused the error.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.18.1.1.2 Properties

Properties of the **EIBCErr** class.

For a complete list of the **EIBCErr** class members, see the [EIBCErr Members](#) topic.

Public

Name	Description
Component (inherited from EDAError)	Contains the component that caused the error.
ErrorCode (inherited from EDAError)	Determines the error code returned by the server.
ErrorNumber	Used to determine the error number returned by InterBase.
Sender	Holds the reference to the sender if exception is raised by the TComponent instance.
SQLErrorMsg	Holds the error message describing the part of the SQL code that caused the error.

See Also

- [EIBCErr Class](#)
- [EIBCErr Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.18.1.1.2.1 ErrorNumber Property

Used to determine the error number returned by InterBase.

Class

[EIBCErr](#)

Syntax

```
property ErrorNumber: integer;
```

Remarks

Use the ErrorNumber property to determine the error number returned by InterBase.

See Also

- [EDAError.ErrorCode](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.18.1.1.2.2 Sender Property

Holds the reference to the sender if exception is raised by the TComponent instance.

Class

[EIBError](#)

Syntax

```
property Sender: TComponent;
```

Remarks

The Sender property holds the reference to the sender if exception is raised by the TComponent instance.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.18.1.1.2.3 SQLErrorMsg Property

Holds the error message describing the part of the SQL code that caused the error.

Class

[EIBError](#)

Syntax

```
property SQLErrorMsg: string;
```

Remarks

The error message describing the part of the SQL code that caused the error.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19 IBCLoader

This unit contains implementation of the TIBCLoader component.

Classes

Name	Description
TIBCLoader	This component serves for loading external data into the database table.
TIBCLoaderOptions	This class allows setting up the behaviour of the TIBCLoader class.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1 Classes

Classes in the **IBCLoader** unit.

Classes

Name	Description
TIBCLoader	This component serves for loading external data into the database table.
TIBCLoaderOptions	This class allows setting up the behaviour of the TIBCLoader class.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.1 TIBCLoader Class

This component serves for loading external data into the database table.

For a list of all members of this type, see [TIBCLoader](#) members.

Unit

[IBCLoader](#)

Syntax

```
TIBCLoader = class(TDALoader);
```

Remarks

The TIBCLoader component allows to load external data into the database table.

TIBCLoader serves for fast loading data to the database. To specify the name of the loading table set the [TIBCLoader.TableName](#) property. Use the [TIBCLoader.Columns](#) property to access individual columns. Write the [TIBCLoader.OnGetColumnData](#) or [TIBCLoader.OnPutData](#) event handlers to read external data and pass it to the database. Call the Load method to start loading data.

TIBCLoader loads data by executing INSERT statements. For Firebird 2.0 and higher several INSERT statements are combined in one EXECUTE BLOCK statement to speed up loading (the number of records that are sent to the server at once is controlled by the [TIBCLoaderOptions.RowsPerBatch](#) property).

The [TIBCLoaderOptions.InsertMode](#) property controls whether the INSERT INTO or UPDATE OR INSERT INTO statement will be used for loading data. Using of UPDATE OR INSERT INTO statements is available in Firebird 2.1 and higher.

Inheritance Hierarchy

[TDALoader](#)

TIBCLoader

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.1.1 Members

[TIBCLoader](#) class overview.

Properties

Name	Description
Columns	Used to access individual columns.
Connection (inherited from TDALoader)	property. Used to specify TCustomDAConnection in which TDALoader will be executed.
Options	This class allows setting up the behaviour of the TIBCLoader class.
TableName	Used to specify the name of the loading table set.

Methods

Name	Description
CreateColumns (inherited from TDALoader)	Creates TDAColumn objects for all fields of the table with the same name as TDALoader.TableName .
Load (inherited from TDALoader)	Starts loading data.
LoadFromDataSet (inherited from TDALoader)	Loads data from the specified dataset.
PutColumnData (inherited from TDALoader)	Overloaded. Puts the value of individual columns.

Events

Name	Description
OnGetColumnData	Used to read external data.
OnProgress (inherited from TDALoader)	Occurs if handling data loading progress of the TDALoader.LoadFromDataSet method is needed.
OnPutData	Used to pass external data to the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.1.2 Properties

Properties of the **TIBCLoader** class.

For a complete list of the **TIBCLoader** class members, see the [TIBCLoader Members](#) topic.

Public

Name	Description
Connection (inherited from TDALoader)	property. Used to specify TCustomDACConnection in which TDALoader will be executed.

Published

Name	Description
Columns	Used to access individual columns.
Options	This class allows setting up the behaviour of the TIBCLoader class.
TableName	Used to specify the name of the loading table set.

See Also

- [TIBCLoader Class](#)
- [TIBCLoader Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.1.2.1 Columns Property

Used to access individual columns.

Class

[TIBCLoader](#)

Syntax

```
property Columns: TDAColumns;
```

Remarks

Use the Columns property to access individual columns.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.1.2.2 Options Property

This class allows setting up the behaviour of the TIBCLoader class.

Class

[TIBCLoader](#)

Syntax

```
property Options: TIBCLoaderOptions;
```

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.1.2.3 TableName Property

Used to specify the name of the loading table set.

Class

[TIBCLoader](#)

Syntax

```
property TableName: string;
```

Remarks

Use the TableName property to specify the name of the loading table set.

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.19.1.1.3 Events

Events of the **TIBCLoader** class.

For a complete list of the **TIBCLoader** class members, see the [TIBCLoader Members](#) topic.

Public

Name	Description
OnProgress (inherited from TDALoader)	Occurs if handling data loading progress of the TDALoader.LoadFromDataSet method is needed.

Published

Name	Description
OnGetColumnData	Used to read external data.
OnPutData	Used to pass external data to the database.

See Also

- [TIBCLoader Class](#)
- [TIBCLoader Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.1.3.1 OnGetColumnData Event

Used to read external data.

Class

[TIBCLoader](#)

Syntax

```
property OnGetColumnData: TGetColumnDataEvent;
```

Remarks

Write the OnGetColumnData event handler to read external data.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.1.3.2 OnPutData Event

Used to pass external data to the database.

Class

[TIBCLoader](#)

Syntax

```
property OnPutData: TDAPutDataEvent;
```

Remarks

Write the OnPutData event handler to pass external data to the database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.2 TIBCLoaderOptions Class

This class allows setting up the behaviour of the [TIBCLoader](#) class.

For a list of all members of this type, see [TIBCLoaderOptions](#) members.

Unit

[IBCLoader](#)

Syntax

```
TIBCLoaderOptions = class(TDALoaderOptions);
```

Inheritance Hierarchy

[TDALoaderOptions](#)

TIBCLoaderOptions

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.2.1 Members

[TIBCLoaderOptions](#) class overview.

Properties

Name	Description
InsertMode	Used to control the type of statement used for loading data.
QuoteNames	Used to quote columns names in the INSERT statement used for loading data.
RowsPerBatch	Used to control the number of records that are sent to the server at once.
UseBlankValues (inherited from TDALoaderOptions)	Forces IBDAC to fill the buffer with null values after loading a row to the database.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.2.2 Properties

Properties of the **TIBCLoaderOptions** class.

For a complete list of the **TIBCLoaderOptions** class members, see the [TIBCLoaderOptions Members](#) topic.

Public

Name	Description
UseBlankValues (inherited from TDALoaderOptions)	Forces IBDAC to fill the buffer with null values after loading a row to the database.

Published

Name	Description
InsertMode	Used to control the type of statement used for loading data.
QuoteNames	Used to quote columns names in the INSERT statement used for loading data.
RowsPerBatch	Used to control the number of records that are sent to the server at once.

See Also

- [TIBCLoaderOptions Class](#)
- [TIBCLoaderOptions Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.2.2.1 InsertMode Property

Used to control the type of statement used for loading data.

Class

[TIBCLoaderOptions](#)

Syntax

```
property InsertMode: TIBCInsertMode default imInsert;
```

Remarks

Use the InsertMode property to specify the type of statement used for loading data using the [TIBCLoader](#) component.

When the property value is *imInsert* , then the INSERT INTO statement is used. When the property value is *imUpdateOrInsert* , then the UPDATE OR INSERT INTO statement is used.

Using of UPDATE OR INSERT INTO statements is available in Firebird 2.1 and higher.

The default value is *imInsert* .

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.2.2.2 QuoteNames Property

Used to quote columns names in the INSERT statement used for loading data.

Class

[TIBCLoaderOptions](#)

Syntax

```
property QuoteNames: boolean;
```

Remarks

Use the QuoteNames property to specify whether the column names in the INSERT statement will be quoted.

When the property value is *True* , then the column names will be quoted. When the property value is *False* then the column names will not be quoted.

The default value is *False* .

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.19.1.2.2.3 RowsPerBatch Property

Used to control the number of records that are sent to the server at once.

Class

[TIBCLoaderOptions](#)

Syntax

```
property RowsPerBatch: integer default 50;
```

Remarks

Use the RowsPerBatch property to specify the number of records that are sent to the server at once when loading data using the [TIBCLoader](#) component.

[TIBCLoader](#) loads data by executing INSERT statements. For Firebird 2.0 and higher several INSERT statements (depending on the RowsPerBatch property value) are combined in one EXECUTE BLOCK statement to speed up loading.

The default value is 50.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.20 IBCScript

This unit contains implementation of the TIBCScript component.

Classes

Name	Description
TIBCScript	A component for executing several SQL statements one by one.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.20.1 Classes

Classes in the **IBCScript** unit.

Classes

Name	Description
TIBCScript	A component for executing several SQL statements one by one.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.20.1.1 TIBCScript Class

A component for executing several SQL statements one by one.

For a list of all members of this type, see [TIBCScript](#) members.

Unit

[TIBCScript](#)

Syntax

```
TIBCScript = class(TDAScript);
```

Remarks

Often it is necessary to execute several SQL statements one by one. Known way is using a lot of components such as [TIBCSQL](#). Usually it is not a good solution. With only one TIBCScript component you can execute several SQL statements as one. This sequence of statements is named script. To separate single statements use semicolon (;), slash (/), and for statements that can contain semicolon (for example CREATE TRIGGER or CREATE PROCEDURE) - only slash . Note that slash must be the first character in line.

Errors that occur while execution can be processed in the [TDAScript.OnError](#) event handler. By default, on error TIBCScript shows exception and continues execution.

Inheritance Hierarchy

[TDAScript](#)

TIBCScript

See Also

- [TIBCSQL](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.20.1.1.1 Members

[TIBCScript](#) class overview.

Properties

Name	Description
AutoDDL	Used to indicate whether DDL statements must be executed in a separate transaction.
Connection	Used to specify the connection in which the script will be executed.
DataSet	Used to get or set script result dataset.
Debug (inherited from TDAScript)	Used to display the script execution and all its parameter values.
Delimiter (inherited from TDAScript)	Used to set the delimiter string that separates script statements.
EndLine (inherited from TDAScript)	Used to get the current statement last line number in a script.
EndOffset (inherited from TDAScript)	Used to get the offset in the last line of the current statement.
EndPos (inherited from TDAScript)	Used to get the end position of the current statement.
Macros (inherited from TDAScript)	Used to change SQL script text in design- or run-time easily.
Params	Used to hold the parameters for a SQL script.
SQL (inherited from TDAScript)	Used to get or set script text.
StartLine (inherited from TDAScript)	Used to get the current statement start line number in a script.
StartOffset (inherited from TDAScript)	Used to get the offset in the first line of the current statement.
StartPos (inherited from TDAScript)	Used to get the start position of the current statement in a script.
Statements (inherited from TDAScript)	Contains a list of statements obtained from the SQL property.
Transaction	Used to set or return the

	transaction to be used by the component.
--	--

Methods

Name	Description
BreakExec (inherited from TDAScript)	Stops script execution.
ErrorOffset (inherited from TDAScript)	Used to get the offset of the statement if the Execute method raised an exception.
Execute (inherited from TDAScript)	Executes a script.
ExecuteFile (inherited from TDAScript)	Executes SQL statements contained in a file.
ExecuteNext (inherited from TDAScript)	Executes the next statement in the script and then stops.
ExecuteStream (inherited from TDAScript)	Executes SQL statements contained in a stream object.
FindMacro (inherited from TDAScript)	Finds a macro with the specified name.
MacroByName (inherited from TDAScript)	Finds a macro with the specified name.

Events

Name	Description
AfterExecute (inherited from TDAScript)	Occurs after a SQL script execution.
BeforeExecute (inherited from TDAScript)	Occurs when taking a specific action before executing the current SQL statement is needed.
OnError (inherited from TDAScript)	Occurs when InterBase raises an error.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.20.1.1.2 Properties

Properties of the **TIBCScript** class.

For a complete list of the **TIBCScript** class members, see the [TIBCScript Members](#) topic.

Public

Name	Description
EndLine (inherited from TDAScript)	Used to get the current statement last line number in a script.
EndOffset (inherited from TDAScript)	Used to get the offset in the last line of the current statement.
EndPos (inherited from TDAScript)	Used to get the end position of the current statement.
Params	Used to hold the parameters for a SQL script.
StartLine (inherited from TDAScript)	Used to get the current statement start line number in a script.
StartOffset (inherited from TDAScript)	Used to get the offset in the first line of the current statement.
StartPos (inherited from TDAScript)	Used to get the start position of the current statement in a script.
Statements (inherited from TDAScript)	Contains a list of statements obtained from the SQL property.

Published

Name	Description
AutoDDL	Used to indicate whether DDL statements must be executed in a separate transaction.
Connection	Used to specify the connection in which the script will be executed.
DataSet	Used to get or set script result dataset.

Debug (inherited from TDAScript)	Used to display the script execution and all its parameter values.
Delimiter (inherited from TDAScript)	Used to set the delimiter string that separates script statements.
Macros (inherited from TDAScript)	Used to change SQL script text in design- or run-time easily.
SQL (inherited from TDAScript)	Used to get or set script text.
Transaction	Used to set or return the transaction to be used by the component.

See Also

- [TIBCScript Class](#)
- [TIBCScript Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.20.1.1.2.1 AutoDDL Property

Used to indicate whether DDL statements must be executed in a separate transaction.

Class

[TIBCScript](#)

Syntax

```
property AutoDDL: Boolean default True;
```

Remarks

Use the AutoDDL property to determine whether DDL statements must be executed in a separate transaction.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.20.1.1.2.2 Connection Property

Used to specify the connection in which the script will be executed.

Class

[TIBCScript](#)

Syntax

```
property Connection: TIBConnection;
```

Remarks

Use the Connection property to specify the connection in which the script will be executed. If a connection has not been opened before running the Execute method, the TIBConnection.Connect method is called.

See Also

- [TIBConnection](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.20.1.1.2.3 DataSet Property

Used to get or set script result dataset.

Class

[TIBCScript](#)

Syntax

```
property DataSet: TCustomIBCDataset;
```

Remarks

Use the DataSet property to get or set script result dataset.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.20.1.1.2.4 Params Property

Used to hold the parameters for a SQL script.

Class

[TIBCScript](#)

Syntax

```
property Params : TIBCPParams ;
```

Remarks

Contains the parameters for a SQL script.

Access Params at runtime to view and set parameter names, values, and data types dynamically (at design time use the Parameters editor to set parameter information). Params is a zero-based array of parameter records. Index specifies the array element to access.

An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName.

See Also

- [TIBCPParams](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.20.1.1.2.5 Transaction Property

Used to set or return the transaction to be used by the component.

Class

[TIBCScript](#)

Syntax

```
property Transaction : TIBCTransaction stored IsTransactionStored;
```

Remarks

Use the Transaction property to set or return the transaction to be used by the component.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.21 IBCSQLMonitor

This unit contains implementation of the TIBCSQLMonitor component.

Classes

Name	Description
TIBCSQLMonitor	[Inherited]

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.21.1 Classes

Classes in the **IBCSQLMonitor** unit.

Classes

Name	Description
TIBCSQLMonitor	[Inherited]

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.21.1.1 TIBCSQLMonitor Class

[Inherited]

For a list of all members of this type, see [TIBCSQLMonitor](#) members.

Unit

[IBCSQLMonitor](#)

Syntax

```
TIBCSQLMonitor = class(TCustomDASQLMonitor);
```

Remarks

Use TIBCSQLMonitor to monitor dynamic SQL execution in IBDAC-based applications. TIBCSQLMonitor provides two ways of displaying debug information: with dialog window, [DBMonitor](#) or Borland SQL Monitor. Furthermore to receive debug information the [TCustomDASQLMonitor.OnSQL](#) event can be used. Also it is possible to use all these ways at the same time, though an application may have only one TIBCSQLMonitor object. If an application has no TIBCSQLMonitor instance, the Debug window is available to display SQL statements to be sent.

Inheritance Hierarchy

[TCustomDASQLMonitor](#)

TIBCSQLMonitor

See Also

- [TCustomDADataset.Debug](#)
- [TCustomDASQL.Debug](#)
- [DBMonitor](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.21.1.1.1 Members

[TIBCSQLMonitor](#) class overview.

Properties

Name	Description
Active (inherited from TCustomDASQLMonitor)	Used to activate monitoring of SQL.
DBMonitorOptions (inherited from TCustomDASQLMonitor)	Used to set options for dbMonitor.
Options (inherited from TCustomDASQLMonitor)	Used to include the desired properties for TCustomDASQLMonitor.
TraceFlags (inherited from TCustomDASQLMonitor)	Used to specify which database operations the monitor should track in an

	application at runtime.
--	-------------------------

Events

Name	Description
OnSQL (inherited from TCustomDASQLMonitor)	Occurs when tracing of SQL activity on database components is needed.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.22 IbDacVcl

This unit contains the visual constituent of IBDAC.

Classes

Name	Description
TIBConnectDialog	A class that provides a dialog box for user to supply his login information.

Routines

Name	Description
GetIBCDatabaseList	Loads database name history from the registry.
GetIBCDatabaseList	Loads database name history from the registry.
GetIBCServerList	Loads server name history from the registry.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.22.1 Classes

Classes in the **IbDacVcl** unit.

Classes

Name	Description
TIBConnectDialog	A class that provides a dialog box for user to supply his login information.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.22.1.1 TIBConnectDialog Class

A class that provides a dialog box for user to supply his login information.

For a list of all members of this type, see [TIBConnectDialog](#) members.

Unit

[IbDacvc1](#)

Syntax

```
TIBConnectDialog = class(TCustomConnectDialog);
```

Remarks

The TIBConnectDialog component is a direct descendant of TCustomConnectDialog class. Use TIBConnectDialog to provide dialog box for user to supply server name, user name, and password. You may want to customize appearance of dialog box using this class's properties.

Inheritance Hierarchy

[TCustomConnectDialog](#)

TIBConnectDialog

See Also

- [TCustomDAConnection.ConnectDialog](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.22.1.1.1 Members

[TIBConnectDialog](#) class overview.

Properties

Name	Description
CancelButton (inherited from TCustomConnectDialog)	Used to specify the label for the Cancel button.
Caption (inherited from TCustomConnectDialog)	Used to set the caption of dialog box.
ConnectButton (inherited from TCustomConnectDialog)	Used to specify the label for the Connect button.
Connection	Used to indicate the TIBConnection component using the TIBConnectDialog object.
DatabaseLabel	Used to specify a prompt for database edit.
DialogClass (inherited from TCustomConnectDialog)	Used to specify the class of the form that will be displayed to enter login information.
LabelSet (inherited from TCustomConnectDialog)	Used to set the language of buttons and labels captions.
PasswordLabel (inherited from TCustomConnectDialog)	Used to specify a prompt for password edit.
ProtocolLabel	Used to specify a prompt for protocol box.
Retries (inherited from TCustomConnectDialog)	Used to indicate the number of retries of failed connections.
SavePassword (inherited from TCustomConnectDialog)	Used for the password to be displayed in ConnectDialog in asterisks.
ServerLabel (inherited from TCustomConnectDialog)	Used to specify a prompt for the server name edit.
StoreLogInfo (inherited from TCustomConnectDialog)	Used to specify whether the login information should be kept in system registry after a connection was established.

UsernameLabel (inherited from TCustomConnectDialog)	Used to specify a prompt for username edit.
--	---

Methods

Name	Description
Execute (inherited from TCustomConnectDialog)	Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button.
GetServerList (inherited from TCustomConnectDialog)	Retrieves a list of available server names.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.22.1.1.2 Properties

Properties of the **TIBConnectDialog** class.

For a complete list of the **TIBConnectDialog** class members, see the [TIBConnectDialog Members](#) topic.

Public

Name	Description
CancelButton (inherited from TCustomConnectDialog)	Used to specify the label for the Cancel button.
Caption (inherited from TCustomConnectDialog)	Used to set the caption of dialog box.
ConnectButton (inherited from TCustomConnectDialog)	Used to specify the label for the Connect button.
Connection	Used to indicate the TIBConnection component using the TIBConnectDialog object.
DatabaseLabel	Used to specify a prompt for database edit.
DialogClass (inherited from TCustomConnectDialog)	Used to specify the class of the form that will be displayed to enter login

	information.
LabelSet (inherited from TCustomConnectDialog)	Used to set the language of buttons and labels captions.
PasswordLabel (inherited from TCustomConnectDialog)	Used to specify a prompt for password edit.
ProtocolLabel	Used to specify a prompt for protocol box.
Retries (inherited from TCustomConnectDialog)	Used to indicate the number of retries of failed connections.
SavePassword (inherited from TCustomConnectDialog)	Used for the password to be displayed in ConnectDialog in asterisks.
ServerLabel (inherited from TCustomConnectDialog)	Used to specify a prompt for the server name edit.
StoreLogInfo (inherited from TCustomConnectDialog)	Used to specify whether the login information should be kept in system registry after a connection was established.
UsernameLabel (inherited from TCustomConnectDialog)	Used to specify a prompt for username edit.

See Also

- [TIBConnectDialog Class](#)
- [TIBConnectDialog Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.22.1.1.2.1 Connection Property

Used to indicate the TIBConnection component using the TIBConnectDialog object.

Class

[TIBConnectDialog](#)

Syntax

```
property Connection: TIBConnection;
```

Remarks

Read the Connection property to learn what TIBConnection component uses the TIBConnectDialog object. This property is read only.

See Also

- [TCustomDAConnection.ConnectDialog](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.22.1.1.2.2 DatabaseLabel Property

Used to specify a prompt for database edit.

Class

[TIBConnectDialog](#)

Syntax

```
property DatabaseLabel: string;
```

Remarks

Use the DatabaseLabel property to specify a prompt for database edit.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.22.1.1.2.3 ProtocolLabel Property

Used to specify a prompt for protocol box.

Class

[TIBConnectDialog](#)

Syntax

```
property ProtocolLabel: string;
```

Remarks

Use the ProtocolLabel property to specify a prompt for protocol box.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.22.2 Routines

Routines in the **IbDacVcl** unit.

Routines

Name	Description
GetIBCDatabaseList	Loads database name history from the registry.
GetIBCDatabaseList	Loads database name history from the registry.
GetIBCServerList	Loads server name history from the registry.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.22.2.1 GetIBCDatabaseList Procedure

Loads database name history from the registry.

Unit

[IbDacVcl](#)

Syntax

```
procedure GetIBCDatabaseList(Server: string; List: TStrings);
overload;
```

Parameters

Server

Specifies the server name for which the database name history will be loaded.

List

Defines the string list, into which the database name list will be populated.

Remarks

In addition, the `GetIBCDatabaseList` procedure can be used when building a custom connection dialog as it is shown in the IBDAC demo.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.22.2.2 `GetIBCDatabaseList` Procedure

Loads database name history from the registry.

Unit

[IbDacVc1](#)

Syntax

```
procedure GetIBCDatabaseList(ServerIndex: integer; List: TStrings); overload;
```

Parameters

ServerIndex

Specifies the index of the server name in the server list (starts from 1), for which the database name history will be loaded.

List

Defines the string list, into which the database name list will be populated.

Remarks

In addition, the `GetIBCDatabaseList` procedure can be used when building a custom connection dialog as it is shown in the IBDAC demo.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.22.2.3 `GetIBCServerList` Procedure

Loads server name history from the registry.

Unit

[IbDacVc1](#)

Syntax

```
procedure GetIBCServerList(List: TStrings; withEmpties: boolean =
False);
```

Parameters

List

Defines the string list into which the server name list will be populated

WithEmpties

Specifies whether blank server names will be inserted into the list. The default value is False.

Remarks

In addition, the GetIBCServerList procedure can be used when building a custom connection dialog as it is shown in the IBDAC demo.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23 MemData

This unit contains classes for storing data in memory.

Classes

Name	Description
TAttribute	TAttribute is not used in IBDAC.
TBlob	Holds large object value for field and parameter dtBlob, dtMemo data types.
TCompressedBlob	Holds large object value for field and parameter dtBlob, dtMemo data types and can compress its data.
TDBObject	A base class for classes that work with user-defined data types that have attributes.
TMemData	Implements in-memory database.
TObjectType	This class is not used.

TSharedObject	A base class that allows to simplify memory management for object referenced by several other objects.
-------------------------------	--

Types

Name	Description
TLocateExOptions	Represents the set of TLocateExOption .
TUpdateRecKinds	Represents the set of TUpdateRecKind.

Enumerations

Name	Description
TCompressBlobMode	Specifies when the values should be compressed and the way they should be stored.
TConnLostCause	Specifies the cause of the connection loss.
TDANumericType	Specifies the format of storing and representing of the NUMERIC (DECIMAL) fields.
TLocateExOption	Allows to set additional search parameters which will be used by the LocateEx method.
TSortType	Specifies a sort type for string fields.
TUpdateRecKind	Indicates records for which the ApplyUpdates method will be performed.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1 Classes

Classes in the **MemData** unit.

Classes

Name	Description
TAttribute	TAttribute is not used in IBDAC.
TBlob	Holds large object value for field and parameter dtBlob, dtMemo data types.
TCompressedBlob	Holds large object value for field and parameter dtBlob, dtMemo data types and can compress its data.
TDBObject	A base class for classes that work with user-defined data types that have attributes.
TMemData	Implements in-memory database.
TObjectType	This class is not used.
TSharedObject	A base class that allows to simplify memory management for object referenced by several other objects.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.1 TAttribute Class

TAttribute is not used in IBDAC.

For a list of all members of this type, see [TAttribute](#) members.

Unit

[MemData](#)

Syntax

```
TAttribute = class(System.TObject);
```

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.1.1 Members

[TAttribute](#) class overview.

Properties

Name	Description
AttributeNo	Returns an attribute's ordinal position in object.
DataSize	Returns the size of an attribute value in internal representation.
DataType	Returns the type of data that was assigned to the Attribute.
Length	Returns the length of the string for dtString attribute and precision for dtInteger and dtFloat attribute.
ObjectType	Returns a TObjectType object for an object attribute.
Offset	Returns an offset of the attribute value in internal representation.
Owner	Indicates TObjectType that uses the attribute to represent one of its attributes.
Scale	Returns the scale of dtFloat and dtInteger attributes.
Size	Returns the size of an attribute value in external representation.

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.1.2 Properties

Properties of the **TAttribute** class.

For a complete list of the **TAttribute** class members, see the [TAttribute Members](#) topic.

Public

Name	Description
AttributeNo	Returns an attribute's ordinal position in object.
DataSize	Returns the size of an attribute value in internal representation.
DataType	Returns the type of data that was assigned to the Attribute.
Length	Returns the length of the string for dtString attribute and precision for dtInteger and dtFloat attribute.
ObjectType	Returns a TObjectType object for an object attribute.
Offset	Returns an offset of the attribute value in internal representation.
Owner	Indicates TObjectType that uses the attribute to represent one of its attributes.
Scale	Returns the scale of dtFloat and dtInteger attributes.
Size	Returns the size of an attribute value in external representation.

See Also

- [TAttribute Class](#)
- [TAttribute Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.1.2.1 AttributeNo Property

Returns an attribute's ordinal position in object.

Class

[TAttribute](#)

Syntax

```
property AttributeNo: word;
```

Remarks

Use the AttributeNo property to learn an attribute's ordinal position in object, where 1 is the first field.

See Also

- [TObjectType.Attributes](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.1.2.2 DataSize Property

Returns the size of an attribute value in internal representation.

Class

[TAttribute](#)

Syntax

```
property DataSize: Integer;
```

Remarks

Use the DataSize property to learn the size of an attribute value in internal representation.

For example:

dtDate	17 (sizeof(OCIDate))
dtFloat	22

	(sizeof(OCINumber))
dtInteger	22 (sizeof(OCINumber))

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.1.2.3 DataType Property

Returns the type of data that was assigned to the Attribute.

Class

[TAttribute](#)

Syntax

```
property DataType: word;
```

Remarks

Use the DataType property to discover the type of data that was assigned to the Attribute.

Possible values: dtDate, dtFloat, dtInteger, dtString, dtObject.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.1.2.4 Length Property

Returns the length of the string for dtString attribute and precision for dtInteger and dtFloat attribute.

Class

[TAttribute](#)

Syntax

```
property Length: word;
```

Remarks

Use the Length property to learn the length of the string for dtString attribute and precision for dtInteger and dtFloat attribute.

See Also

- [Scale](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.1.2.5 ObjectType Property

Returns a TObjectType object for an object attribute.

Class

[TAttribute](#)

Syntax

```
property objectType: TObjectType;
```

Remarks

Use the ObjectType property to return a TObjectType object for an object attribute.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.1.2.6 Offset Property

Returns an offset of the attribute value in internal representation.

Class

[TAttribute](#)

Syntax

```
property offset: Integer;
```

Remarks

Use the DataSize property to learn an offset of the attribute value in internal representation.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.1.2.7 Owner Property

Indicates TObjectType that uses the attribute to represent one of its attributes.

Class

[TAttribute](#)

Syntax

```
property Owner: TObjectType;
```

Remarks

Check the value of the Owner property to determine TObjectType that uses the attribute to represent one of its attributes. Applications should not assign the Owner property directly.

It is assigned automatically when the attribute is created from a TOraType.Describe.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.1.2.8 Scale Property

Returns the scale of dtFloat and dtInteger attributes.

Class

[TAttribute](#)

Syntax

```
property Scale: word;
```

Remarks

Use the Scale property to learn the scale of dtFloat and dtInteger attributes.

See Also

- [Length](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.1.2.9 Size Property

Returns the size of an attribute value in external representation.

Class

[TAttribute](#)

Syntax

```
property Size: Integer;
```

Remarks

Read Size to learn the size of an attribute value in external representation.

For example:

dtDate	8 (sizeof(TDateTi me))
dtFloat	8 (sizeof(Double))
dtInteger	4 (sizeof(Integer))

See Also

- [DataSize](#)

© 1997-2024
Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2 TBlob Class

Holds large object value for field and parameter dtBlob, dtMemo data types.

For a list of all members of this type, see [TBlob](#) members.

Unit

[MemData](#)

Syntax

```
TBlob = class(TSharedObject);
```

Remarks

Object TBlob holds large object value for the field and parameter dtBlob, dtMemo, dtWideMemo data types.

Inheritance Hierarchy

[TSharedObject](#)

TBlob

See Also

- [TMemDataSet.GetBlob](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.1 Members

[TBlob](#) class overview.

Properties

Name	Description
AsString	Used to manipulate BLOB value as string.
AsWideString	Used to manipulate BLOB value as Unicode string.
IsUnicode	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	Used to learn the size of the TBlob value in bytes.

Methods

Name	Description
AddRef (inherited from TSharedObject)	Increments the reference count for the number of references dependent on the TSharedObject object.
Assign	Sets BLOB value from another TBlob object.
Clear	Deletes the current value in TBlob object.
LoadFromFile	Loads the contents of a file into a TBlob object.
LoadFromStream	Copies the contents of a stream into the TBlob object.
Read	Acquires a raw sequence of bytes from the data stored in TBlob.
Release (inherited from TSharedObject)	Decrements the reference count.
SaveToFile	Saves the contents of the TBlob object to a file.
SaveToStream	Copies the contents of a TBlob object to a stream.
Truncate	Sets new TBlob size and discards all data over it.
Write	Stores a raw sequence of bytes into a TBlob object.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.2 Properties

Properties of the **TBlob** class.

For a complete list of the **TBlob** class members, see the [TBlob Members](#) topic.

Public

Name	Description
AsString	Used to manipulate BLOB value as string.
AsWideString	Used to manipulate BLOB value as Unicode string.

IsUnicode	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	Used to learn the size of the TBlob value in bytes.

See Also

- [TBlob Class](#)
- [TBlob Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.2.1 AsString Property

Used to manipulate BLOB value as string.

Class

[TBlob](#)

Syntax

```
property AsString: string;
```

Remarks

Use the AsString property to manipulate BLOB value as string.

See Also

- [Assign](#)
- [AsWideString](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.2.2 AsWideString Property

Used to manipulate BLOB value as Unicode string.

Class

[TBlob](#)

Syntax

```
property AsWideString: string;
```

Remarks

Use the AsWideString property to manipulate BLOB value as Unicode string.

See Also

- [Assign](#)
- [AsString](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.2.3 IsUnicode Property

Gives choice of making TBlob store and process data in Unicode format or not.

Class

[TBlob](#)

Syntax

```
property IsUnicode: boolean;
```

Remarks

Set IsUnicode to True if you want TBlob to store and process data in Unicode format.

Note: changing this property raises an exception if TBlob is not empty.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.2.4 Size Property

Used to learn the size of the TBlob value in bytes.

Class

[TBlob](#)

Syntax

```
property size: cardinal;
```

Remarks

Use the Size property to find out the size of the TBlob value in bytes.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.3 Methods

Methods of the **TBlob** class.

For a complete list of the **TBlob** class members, see the [TBlob Members](#) topic.

Public

Name	Description
AddRef (inherited from TSharedObject)	Increments the reference count for the number of references dependent on the TSharedObject object.
Assign	Sets BLOB value from another TBlob object.
Clear	Deletes the current value in TBlob object.
LoadFromFile	Loads the contents of a file into a TBlob object.
LoadFromStream	Copies the contents of a stream into the TBlob object.
Read	Acquires a raw sequence of bytes from the data stored in TBlob.
Release (inherited from TSharedObject)	Decrements the reference count.

SaveToFile	Saves the contents of the TBlob object to a file.
SaveToStream	Copies the contents of a TBlob object to a stream.
Truncate	Sets new TBlob size and discards all data over it.
Write	Stores a raw sequence of bytes into a TBlob object.

See Also

- [TBlob Class](#)
- [TBlob Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.3.1 Assign Method

Sets BLOB value from another TBlob object.

Class

[TBlob](#)

Syntax

```
procedure Assign(Source: TBlob);
```

Parameters

Source

Holds the BLOB from which the value to the current object will be assigned.

Remarks

Call the Assign method to set BLOB value from another TBlob object.

See Also

- [LoadFromStream](#)
- [AsString](#)
- [AsWideString](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.3.2 Clear Method

Deletes the current value in TBlob object.

Class

[TBlob](#)

Syntax

```
procedure Clear; virtual;
```

Remarks

Call the Clear method to delete the current value in TBlob object.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.3.3 LoadFromFile Method

Loads the contents of a file into a TBlob object.

Class

[TBlob](#)

Syntax

```
procedure LoadFromFile(const FileName: string);
```

Parameters

FileName

Holds the name of the file from which the TBlob value is loaded.

Remarks

Call the LoadFromFile method to load the contents of a file into a TBlob object. Specify the name of the file to load into the field as the value of the FileName parameter.

See Also

- [SaveToFile](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.3.4 LoadFromStream Method

Copies the contents of a stream into the TBlob object.

Class

[TBlob](#)

Syntax

```
procedure LoadFromStream(Stream: TStream); virtual;
```

Parameters

Stream

Holds the specified stream from which the field's value is copied.

Remarks

Call the LoadFromStream method to copy the contents of a stream into the TBlob object. Specify the stream from which the field's value is copied as the value of the Stream parameter.

See Also

- [SaveToStream](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.3.5 Read Method

Acquires a raw sequence of bytes from the data stored in TBlob.

Class

[TBlob](#)

Syntax

```
function Read(Position: Cardinal; Count: Cardinal; Dest: IntPtr):  
Cardinal; virtual;
```

Parameters*Position*

Holds the starting point of the byte sequence.

Count

Holds the size of the sequence in bytes.

Dest

Holds a pointer to the memory area where to store the sequence.

Return Value

Actually read byte count if the sequence crosses object size limit.

Remarks

Call the Read method to acquire a raw sequence of bytes from the data stored in TBlob.

The Position parameter is the starting point of byte sequence which lasts Count number of bytes. The Dest parameter is a pointer to the memory area where to store the sequence.

If the sequence crosses object size limit, function will return actually read byte count.

See Also

- [Write](#)

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.3.6 SaveToFile Method

Saves the contents of the TBlob object to a file.

Class

[TBlob](#)

Syntax

```
procedure SaveToFile(const FileName: string);
```

Parameters*FileName*

Holds a string that contains the name of the file.

Remarks

Call the `SaveToFile` method to save the contents of the `TBlob` object to a file. Specify the name of the file as the value of the `FileName` parameter.

See Also

- [LoadFromFile](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.3.7 SaveToStream Method

Copies the contents of a `TBlob` object to a stream.

Class

[TBlob](#)

Syntax

```
procedure SaveToStream(Stream: TStream); virtual;
```

Parameters

Stream

Holds the name of the stream.

Remarks

Call the `SaveToStream` method to copy the contents of a `TBlob` object to a stream. Specify the name of the stream to which the field's value is saved as the value of the `Stream` parameter.

See Also

- [LoadFromStream](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.3.8 Truncate Method

Sets new TBlob size and discards all data over it.

Class

[TBlob](#)

Syntax

```
procedure Truncate(NewSize: Cardinal); virtual;
```

Parameters

NewSize

Holds the new size of TBlob.

Remarks

Call the Truncate method to set new TBlob size and discard all data over it. If NewSize is greater or equal TBlob.Size, it does nothing.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.2.3.9 Write Method

Stores a raw sequence of bytes into a TBlob object.

Class

[TBlob](#)

Syntax

```
procedure write(Position: Cardinal; Count: Cardinal; Source: IntPtr); virtual;
```

Parameters

Position

Holds the starting point of the byte sequence.

Count

Holds the size of the sequence in bytes.

Source

Holds a pointer to a source memory area.

Remarks

Call the Write method to store a raw sequence of bytes into a TBlob object.

The Position parameter is the starting point of byte sequence which lasts Count number of bytes. The Source parameter is a pointer to a source memory area.

If the value of the Position parameter crosses current size limit of TBlob object, source data will be appended to the object data.

See Also

- [Read](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.3 TCompressedBlob Class

Holds large object value for field and parameter dtBlob, dtMemo data types and can compress its data.

For a list of all members of this type, see [TCompressedBlob](#) members.

Unit

[MemData](#)

Syntax

```
TCompressedBlob = class(TBlob);
```

Remarks

TCompressedBlob is a descendant of the TBlob class. It holds large object value for field and parameter dtBlob, dtMemo data types and can compress its data. For more information about using BLOB compression see [TCustomDADataset.Options](#).

Note: Internal compression functions are available in CodeGear Delphi 2007 for Win32, Borland Developer Studio 2006, Borland Delphi 2005, and Borland Delphi 7. To use BLOB compression under Borland Delphi 6 and Borland C++ Builder you should use your own compression functions. To use them set the CompressProc and UncompressProc variables declared in the MemUtils unit.

Example

```

type
TCompressProc = function(dest: IntPtr; destLen: IntPtr; const source: IntPtr;
TUncompressProc = function(dest: IntPtr; destLen: IntPtr; source: IntPtr;
var
CompressProc: TCompressProc;
UncompressProc: TUncompressProc;

```

Inheritance Hierarchy

[TSharedObject](#)

[TBlob](#)

TCompressedBlob

See Also

- [TBlob](#)
- [TMemDataSet.GetBlob](#)
- [TCustomDADDataSet.Options](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.3.1 Members

[TCompressedBlob](#) class overview.

Properties

Name	Description
AsString (inherited from TBlob)	Used to manipulate BLOB value as string.
AsWideString (inherited from TBlob)	Used to manipulate BLOB value as Unicode string.
Compressed	Used to indicate if the Blob is compressed.
CompressedSize	Used to indicate compressed size of the Blob data.
IsUnicode (inherited from TBlob)	Gives choice of making TBlob store and process data in Unicode format or

	not.
RefCount (inherited from TSharedObject)	Used to return the count of reference to a TSharedObject object.
Size (inherited from TBlob)	Used to learn the size of the TBlob value in bytes.

Methods

Name	Description
AddRef (inherited from TSharedObject)	Increments the reference count for the number of references dependent on the TSharedObject object.
Assign (inherited from TBlob)	Sets BLOB value from another TBlob object.
Clear (inherited from TBlob)	Deletes the current value in TBlob object.
LoadFromFile (inherited from TBlob)	Loads the contents of a file into a TBlob object.
LoadFromStream (inherited from TBlob)	Copies the contents of a stream into the TBlob object.
Read (inherited from TBlob)	Acquires a raw sequence of bytes from the data stored in TBlob.
Release (inherited from TSharedObject)	Decrements the reference count.
SaveToFile (inherited from TBlob)	Saves the contents of the TBlob object to a file.
SaveToStream (inherited from TBlob)	Copies the contents of a TBlob object to a stream.
Truncate (inherited from TBlob)	Sets new TBlob size and discards all data over it.
Write (inherited from TBlob)	Stores a raw sequence of bytes into a TBlob object.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.3.2 Properties

Properties of the **TCompressedBlob** class.

For a complete list of the **TCompressedBlob** class members, see the [TCompressedBlob Members](#) topic.

Public

Name	Description
AsString (inherited from TBlob)	Used to manipulate BLOB value as string.
AsWideString (inherited from TBlob)	Used to manipulate BLOB value as Unicode string.
Compressed	Used to indicate if the Blob is compressed.
CompressedSize	Used to indicate compressed size of the Blob data.
IsUnicode (inherited from TBlob)	Gives choice of making TBlob store and process data in Unicode format or not.
RefCount (inherited from TSharedObject)	Used to return the count of reference to a TSharedObject object.
Size (inherited from TBlob)	Used to learn the size of the TBlob value in bytes.

See Also

- [TCompressedBlob Class](#)
- [TCompressedBlob Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.3.2.1 Compressed Property

Used to indicate if the Blob is compressed.

Class

[TCompressedBlob](#)

Syntax

```
property Compressed: boolean;
```

Remarks

Indicates whether the Blob is compressed. Set this property to True or False to compress or decompress the Blob.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.3.2.2 CompressedSize Property

Used to indicate compressed size of the Blob data.

Class

[TCompressedBlob](#)

Syntax

```
property CompressedSize: Cardinal;
```

Remarks

Indicates compressed size of the Blob data.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.4 TDBObject Class

A base class for classes that work with user-defined data types that have attributes.

For a list of all members of this type, see [TDBObject](#) members.

Unit

[MemData](#)

Syntax

```
TDBObject = class(TSharedObject);
```

Remarks

TDBObject is a base class for classes that work with user-defined data types that have attributes.

Inheritance Hierarchy

[TSharedObject](#)

TDBObject

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.4.1 Members

[TDBObject](#) class overview.

Properties

Name	Description
RefCount (inherited from TSharedObject)	Used to return the count of reference to a TSharedObject object.

Methods

Name	Description
AddRef (inherited from TSharedObject)	Increments the reference count for the number of references dependent on the TSharedObject object.
Release (inherited from TSharedObject)	Decrements the reference count.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.5 TMemData Class

Implements in-memory database.

For a list of all members of this type, see [TMemData](#) members.

Unit

[MemData](#)

Syntax

```
TMemData = class(TData);
```

Inheritance Hierarchy

TData

TMemData

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.5.1 Members

[TMemData](#) class overview.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.6 TObjectType Class

This class is not used.

For a list of all members of this type, see [TObjectType](#) members.

Unit

[MemData](#)

Syntax

```
TObjectType = class(TSharedObject);
```

Inheritance Hierarchy

[TSharedObject](#)**TObjectType**

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.23.1.6.1 Members

[TObjectType](#) class overview.

Properties

Name	Description
AttributeCount	Used to indicate the number of attributes of type.
Attributes	Used to access separate attributes.
DataType	Used to indicate the type of object dtObject, dtArray or dtTable.
RefCount (inherited from TSharedObject)	Used to return the count of reference to a TSharedObject object.
Size	Used to learn the size of an object instance.

Methods

Name	Description
AddRef (inherited from TSharedObject)	Increments the reference count for the number of references dependent on the TSharedObject object.
FindAttribute	Indicates whether a specified Attribute component is referenced in the TAttributes object.
Release (inherited from TSharedObject)	Decrements the reference count.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)[DAC Forum](#)[Provide Feedback](#)

5.23.1.6.2 Properties

Properties of the **TObjectType** class.

For a complete list of the **TObjectType** class members, see the [TObjectType Members](#) topic.

Public

Name	Description
AttributeCount	Used to indicate the number of attributes of type.
Attributes	Used to access separate attributes.
DataType	Used to indicate the type of object dtObject, dtArray or dtTable.
RefCount (inherited from TSharedObject)	Used to return the count of reference to a TSharedObject object.
Size	Used to learn the size of an object instance.

See Also

- [TObjectType Class](#)
- [TObjectType Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.6.2.1 AttributeCount Property

Used to indicate the number of attributes of type.

Class

[TObjectType](#)

Syntax

```
property AttributeCount: Integer;
```

Remarks

Use the AttributeCount property to determine the number of attributes of type.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.6.2.2 Attributes Property(Indexer)

Used to access separate attributes.

Class

[TObjectType](#)

Syntax

```
property Attributes[Index: integer]: TAttribute;
```

Parameters

Index

Holds the attribute's ordinal position.

Remarks

Use the Attributes property to access individual attributes. The value of the Index parameter corresponds to the AttributeNo property of TAttribute.

See Also

- [TAttribute](#)
- [FindAttribute](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.6.2.3 DataType Property

Used to indicate the type of object dtObject, dtArray or dtTable.

Class

[TObjectType](#)

Syntax

```
property DataType: word;
```

Remarks

Use the DataType property to determine the type of object dtObject, dtArray or dtTable.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.6.2.4 Size Property

Used to learn the size of an object instance.

Class

[TObjectType](#)

Syntax

```
property Size: Integer;
```

Remarks

Use the Size property to find out the size of an object instance. Size is a sum of all attribute sizes.

See Also

- [TAttribute.Size](#)

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.6.3 Methods

Methods of the **TObjectType** class.

For a complete list of the **TObjectType** class members, see the [TObjectType Members](#) topic.

Public

Name	Description
AddRef (inherited from TSharedObject)	Increments the reference count for the number of references dependent on the TSharedObject object.
FindAttribute	Indicates whether a specified Attribute component is referenced in the TAttributes object.
Release (inherited from TSharedObject)	Decrements the reference count.

See Also

- [TObjectType Class](#)
- [TObjectType Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.6.3.1 FindAttribute Method

Indicates whether a specified Attribute component is referenced in the TAttributes object.

Class

[TObjectType](#)

Syntax

```
function FindAttribute(const Name: string): TAttribute; virtual;
```

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](#)
- [Attributes](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.7 TSharedObject Class

A base class that allows to simplify memory management for object referenced by several other objects.

For a list of all members of this type, see [TSharedObject](#) members.

Unit

[MemData](#)

Syntax

```
TSharedObject = class(System.TObject);
```

Remarks

TSharedObject allows to simplify memory management for object referenced by several other objects. TSharedObject holds a count of references to itself. When any object (referer object) is going to use TSharedObject, it calls the TSharedObject.AddRef method. Referer object has to call the TSharedObject.Release method after using TSharedObject.

See Also

- [TBlob](#)
- [TObjectType](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.7.1 Members

[TSharedObject](#) class overview.

Properties

Name	Description
RefCount	Used to return the count of reference to a TSharedObject object.

Methods

Name	Description
AddRef	Increments the reference count for the number of references dependent on the TSharedObject object.
Release	Decrements the reference count.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.7.2 Properties

Properties of the **TSharedObject** class.

For a complete list of the **TSharedObject** class members, see the [TSharedObject Members](#) topic.

Public

Name	Description
RefCount	Used to return the count of reference to a TSharedObject object.

See Also

- [TSharedObject Class](#)
- [TSharedObject Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.7.2.1 RefCount Property

Used to return the count of reference to a TSharedObject object.

Class

[TSharedObject](#)

Syntax

```
property RefCount: Integer;
```

Remarks

Returns the count of reference to a TSharedObject object.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.7.3 Methods

Methods of the **TSharedObject** class.

For a complete list of the **TSharedObject** class members, see the [TSharedObject Members](#) topic.

Public

Name	Description
AddRef	Increments the reference count for the number of references dependent on the TSharedObject object.
Release	Decrements the reference count.

See Also

- [TSharedObject Class](#)
- [TSharedObject Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.7.3.1 AddRef Method

Increments the reference count for the number of references dependent on the TSharedObject object.

Class

[TSharedObject](#)

Syntax

```
procedure AddRef;
```

Remarks

Increments the reference count for the number of references dependent on the TSharedObject object.

See Also

- [Release](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.1.7.3.2 Release Method

Decrements the reference count.

Class

[TSharedObject](#)

Syntax

```
procedure Release;
```

Remarks

Call the Release method to decrement the reference count. When RefCount is 1, TSharedObject is deleted from memory.

See Also

- [AddRef](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.2 Types

Types in the **MemData** unit.

Types

Name	Description
TLocateExOptions	Represents the set of TLocateExOption .
TUpdateRecKinds	Represents the set of TUpdateRecKind.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.2.1 TLocateExOptions Set

Represents the set of [TLocateExOption](#).

Unit

[MemData](#)

Syntax

```
TLocateExOptions = set of TLocateExOption;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.2.2 TUpdateRecKinds Set

Represents the set of TUpdateRecKind.

Unit

[MemData](#)

Syntax

```
TUpdateRecKinds = set of TUpdateRecKind;
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.3 Enumerations

Enumerations in the **MemData** unit.

Enumerations

Name	Description
TCompressBlobMode	Specifies when the values should be compressed and the way they should be stored.
TConnLostCause	Specifies the cause of the connection loss.
TDANumericType	Specifies the format of storing and representing of the NUMERIC (DECIMAL) fields.
TLocateExOption	Allows to set additional search parameters which will be used by the LocateEx method.
TSortType	Specifies a sort type for string fields.
TUpdateRecKind	Indicates records for which the ApplyUpdates method will be performed.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.3.1 TCompressBlobMode Enumeration

Specifies when the values should be compressed and the way they should be stored.

Unit

[MemData](#)

Syntax

```
TCompressBlobMode = (cbNone, cbClient, cbServer, cbClientServer);
```

Values

Value	Meaning
cbClient	Values are compressed and stored as compressed data at the client side. Before posting data to the server decompression is performed and data at the server side stored in the original form. Allows to reduce used client memory due to increase access time to field values. The time spent on the opening DataSet and executing Post increases.
cbClientServer	Values are compressed and stored in compressed form. Allows to decrease the volume of used memory at client and server sides. Access time to the field values increases as for cbClient. The time spent on opening DataSet and executing Post decreases. Note: On using cbServer or cbClientServer data on the server is stored as compressed. Other applications can add records in uncompressed format but can't read and write already compressed data. If compressed BLOB is partially changed by another application (if signature was not changed), DAC will consider its value as NULL.Blob compression is not applied to Memo fields because of possible cutting.
cbNone	Values not compressed. The default value.
cbServer	Values are compressed before passing to the server and store at the server in compressed form. Allows to decrease database size on the server. Access time to the field values does not change. The time spent on opening DataSet and executing Post usually decreases.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.3.2 TConnLostCause Enumeration

Specifies the cause of the connection loss.

Unit

[MemData](#)

Syntax

```
TConnLostCause = (clUnknown, clExecute, clOpen, clRefresh, clApply, clServiceQuery, clTransStart, clConnectionApply, clConnect);
```

Values

Value	Meaning
clApply	Connection loss detected during DataSet.ApplyUpdates (Reconnect/Reexecute possible).
clConnect	Connection loss detected during connection establishing (Reconnect possible).
clConnectionApply	Connection loss detected during Connection.ApplyUpdates (Reconnect/Reexecute possible).
clExecute	Connection loss detected during SQL execution (Reconnect with exception is possible).
clOpen	Connection loss detected during execution of a SELECT statement (Reconnect with exception possible).
clRefresh	Connection loss detected during query opening (Reconnect/Reexecute possible).
clServiceQuery	Connection loss detected during service information request (Reconnect/Reexecute possible).
clTransStart	Connection loss detected during transaction start (Reconnect/Reexecute possible). clTransStart has less priority then clConnectionApply.
clUnknown	The connection loss reason is unknown.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.3.3 TDANumericType Enumeration

Specifies the format of storing and representing of the NUMERIC (DECIMAL) fields.

Unit

[MemData](#)

Syntax

```
TDANumericType = (ntFloat, ntBCD, ntFmtBCD);
```

Values

Value	Meaning
ntBCD	Data is stored on the client side as currency and represented as TBCDField. This format allows storing data with precision up to 0,0001.

ntFloat	Data stored on the client side is in double format and represented as TFloatField. The default value.
ntFmtBCD	Data is represented as TFMTBCDField. TFMTBCDField gives greater precision and accuracy than TBCDField, but it is slower.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.3.4 TLocateExOption Enumeration

Allows to set additional search parameters which will be used by the LocateEx method.

Unit

[MemData](#)

Syntax

```
TLocateExOption = (lxCaseInsensitive, lxPartialKey, lxNearest, lxNext, lxUp, lxPartialCompare);
```

Values

Value	Meaning
lxCaseInsensitive	Similar to loCaseInsensitive. Key fields and key values are matched without regard to the case.
lxNearest	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.
lxNext	LocateEx searches from the current record.
lxPartialCompare	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'.
lxPartialKey	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'.
lxUp	LocateEx searches from the current record to the first record.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.3.5 TSortType Enumeration

Specifies a sort type for string fields.

Unit

[MemData](#)

Syntax

```
TSortType = (stCaseSensitive, stCaseInsensitive, stBinary);
```

Values

Value	Meaning
stBinary	Sorting by character ordinal values (this comparison is also case sensitive).
stCaseInsensitive	Sorting without case sensitivity.
stCaseSensitive	Sorting with case sensitivity.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.23.3.6 TUpdateReckKind Enumeration

Indicates records for which the ApplyUpdates method will be performed.

Unit

[MemData](#)

Syntax

```
TUpdateReckKind = (ukUpdate, ukInsert, ukDelete);
```

Values

Value	Meaning
ukDelete	ApplyUpdates will be performed for deleted records.
ukInsert	ApplyUpdates will be performed for inserted records.
ukUpdate	ApplyUpdates will be performed for updated records.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Reserved.

5.24 MemDS

This unit contains implementation of the TMemDataSet class.

Classes

Name	Description
TMemDataSet	A base class for working with data and manipulating data in memory.

Variables

Name	Description
DoNotRaiseExcetionOnUaFail	An exception will be raised if the value of the UpdateAction parameter is uaFail.
SendDataSetChangeEventAfterOpen	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.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1 Classes

Classes in the **MemDS** unit.

Classes

Name	Description
TMemDataSet	A base class for working with data and manipulating data in memory.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1 TMemDataSet Class

A base class for working with data and manipulating data in memory.

For a list of all members of this type, see [TMemDataSet](#) members.

Unit

[MemDS](#)

Syntax

```
TMemDataSet = class(TDataSet);
```

Remarks

TMemDataSet derives from the TDataSet database-engine independent set of properties, events, and methods for working with data and introduces additional techniques to store and manipulate data in memory.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.1 Members

[TMemDataSet](#) class overview.

Properties

Name	Description
CachedUpdates	Used to enable or disable the use of cached updates for a dataset.
IndexFieldNames	Used to get or set the list of fields on which the recordset is sorted.
KeyExclusive	Specifies the upper and lower boundaries for a range.
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.
Ranged	Indicates whether a range is applied to a dataset.
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.

Methods

Name	Description
ApplyRange	Applies a range to the dataset.
ApplyUpdates	Overloaded. Writes dataset's pending cached updates to a database.
CancelRange	Removes any ranges currently in effect for a dataset.
CancelUpdates	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates	Clears the cached updates buffer.
DeferredPost	Makes permanent changes to the database server.
EditRangeEnd	Enables changing the ending value for an existing range.
EditRangeStart	Enables changing the starting value for an existing range.
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.
SetRange	Sets the starting and ending values of a range, and applies it.
SetRangeEnd	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
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
OnUpdateError	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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.2 Properties

Properties of the **TMemDataSet** class.

For a complete list of the **TMemDataSet** class members, see the [TMemDataSet Members](#) topic.

Public

Name	Description
CachedUpdates	Used to enable or disable the use of cached updates for a dataset.
IndexFieldNames	Used to get or set the list of fields on which the recordset is sorted.
KeyExclusive	Specifies the upper and lower boundaries for a range.
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.
Ranged	Indicates whether a range is applied to a dataset.
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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.2.1 CachedUpdates Property

Used to enable or disable the use of cached updates for a dataset.

Class

[TMemDataSet](#)

Syntax

```
property cachedUpdates: boolean default False;
```

Remarks

Use the CachedUpdates property to enable or disable the use of cached updates for a dataset. Setting CachedUpdates to True enables updates to a dataset (such as posting changes, inserting new records, or deleting records) to be stored in an internal cache on the client side instead of being written directly to the dataset's underlying database tables. When

changes are completed, an application writes all cached changes to the database in the context of a single transaction.

Cached updates are especially useful for client applications working with remote database servers. Enabling cached updates brings up the following benefits:

- Fewer transactions and shorter transaction times.
- Minimized network traffic.

The potential drawbacks of enabling cached updates are:

- Other applications can access and change the actual data on the server while users are editing local copies of data, resulting in an update conflict when cached updates are applied to the database.
- Other applications cannot access data changes made by an application until its cached updates are applied to the database.

The default value is False.

Note: When establishing master/detail relationship the `CachedUpdates` property of detail dataset works properly only when [TDADatasetOptions.LocalMasterDetail](#) is set to True.

See Also

- [UpdatesPending](#)
- [TMemDataSet.ApplyUpdates](#)
- [RestoreUpdates](#)
- [CommitUpdates](#)
- [CancelUpdates](#)
- [UpdateStatus](#)
- [TCustomDADataset.Options](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.2.2 IndexFieldNames Property

Used to get or set the list of fields on which the recordset is sorted.

Class

[TMemDataSet](#)

Syntax

```
property IndexFieldNames: string;
```

Remarks

Use the IndexFieldNames property to get or set the list of fields on which the recordset is sorted. Specify the name of each column in IndexFieldNames to use as an index for a table. Column names order is significant. Separate names with semicolons. 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 order for the field. If one of these keywords is not used, the default sort order for the field is ascending.

Use CIS, CS or BIN keywords to specify the 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 [TCustomDACConnection](#) component, the default value of the sort type depends on the [TCustomDACConnection.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.

Sorting is performed locally.

Note:

You cannot sort by BLOB fields.

IndexFieldNames cannot be set to True when [TCustomDADataSet.UniDirectional](#)=True.

Example

The following procedure illustrates how to set IndexFieldNames in response to a button click:

```
DataSet1.IndexFieldNames := 'LastName ASC CIS; DateDue DESC';
```

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.2.3 KeyExclusive Property

Specifies the upper and lower boundaries for a range.

Class

[TMemDataSet](#)

Syntax

```
property KeyExclusive: Boolean;
```

Remarks

Use KeyExclusive to specify whether a range includes or excludes the records that match its specified starting and ending values.

By default, KeyExclusive is False, meaning that matching values are included.

To restrict a range to those records that are greater than the specified starting value and less than the specified ending value, set KeyExclusive to True.

See Also

- [SetRange](#)
- [SetRangeEnd](#)
- [SetRangeStart](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.2.4 LocalConstraints Property

Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

Class

[TMemDataSet](#)

Syntax

```
property LocalConstraints: boolean default True;
```

Remarks

Use the LocalConstraints property to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet. When LocalConstraints is True, TMemDataSet ignores NOT NULL server constraints. It is useful for tables that have fields updated by triggers.

LocalConstraints is obsolete, and is only included for backward compatibility.

The default value is True.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.2.5 LocalUpdate Property

Used to prevent implicit update of rows on database server.

Class

[TMemDataSet](#)

Syntax

```
property LocalUpdate: boolean default False;
```

Remarks

Set the LocalUpdate property to True to prevent implicit update of rows on database server. Data changes are cached locally in client memory.

© 1997-2024

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Devart. All Rights Reserved.

5.24.1.1.2.6 Prepared Property

Determines whether a query is prepared for execution or not.

Class

[TMemDataSet](#)

Syntax

```
property Prepared: boolean;
```

Remarks

Determines whether a query is prepared for execution or not.

See Also

- [Prepare](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.2.7 Ranged Property

Indicates whether a range is applied to a dataset.

Class

[TMemDataSet](#)

Syntax

```
property Ranged: Boolean;
```

Remarks

Use the Ranged property to detect whether a range is applied to a dataset.

See Also

- [SetRange](#)

- [SetRangeEnd](#)
- [SetRangeStart](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.2.8 UpdateRecordTypes Property

Used to indicate the update status for the current record when cached updates are enabled.

Class

[TMemDataSet](#)

Syntax

```
property UpdateRecordTypes: TUpdateRecordTypes default  
[rtModified, rtInserted, rtUnmodified];
```

Remarks

Use the UpdateRecordTypes property to determine the update status for the current record when cached updates are enabled. Update status can change frequently as records are edited, inserted, or deleted. UpdateRecordTypes offers a convenient method for applications to assess the current status before undertaking or completing operations that depend on the update status of records.

See Also

- [CachedUpdates](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.2.9 UpdatesPending Property

Used to check the status of the cached updates buffer.

Class

[TMemDataSet](#)

Syntax

```
property UpdatesPending: boolean;
```

Remarks

Use the UpdatesPending property to check the status of the cached updates buffer. If UpdatesPending is True, then there are edited, deleted, or inserted records remaining in local cache and not yet applied to the database. If UpdatesPending is False, there are no such records in the cache.

See Also

- [CachedUpdates](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3 Methods

Methods of the **TMemDataSet** class.

For a complete list of the **TMemDataSet** class members, see the [TMemDataSet Members](#) topic.

Public

Name	Description
ApplyRange	Applies a range to the dataset.
ApplyUpdates	Overloaded. Writes dataset's pending cached updates to a database.
CancelRange	Removes any ranges currently in effect for a dataset.
CancelUpdates	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates	Clears the cached updates buffer.
DeferredPost	Makes permanent changes to the database server.

EditRangeEnd	Enables changing the ending value for an existing range.
EditRangeStart	Enables changing the starting value for an existing range.
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.
SetRange	Sets the starting and ending values of a range, and applies it.
SetRangeEnd	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart	Indicates that subsequent assignments to field values

	specify the start of the range of rows to include in the dataset.
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.

See Also

- [TMemDataSet Class](#)
- [TMemDataSet Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.1 ApplyRange Method

Applies a range to the dataset.

Class

[TMemDataSet](#)

Syntax

```
procedure ApplyRange;
```

Remarks

Call ApplyRange to cause a range established with [SetRangeStart](#) and [SetRangeEnd](#), or [EditRangeStart](#) and [EditRangeEnd](#), to take effect.

When a range is in effect, only those records that fall within the range are available to the

application for viewing and editing.

After a call to `ApplyRange`, the cursor is left on the first record in the range.

See Also

- [CancelRange](#)
- [EditRangeEnd](#)
- [EditRangeStart](#)
- [IndexFieldNames](#)
- [SetRange](#)
- [SetRangeEnd](#)
- [SetRangeStart](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.2 ApplyUpdates Method

Writes dataset's pending cached updates to a database.

Class

[TMemDataSet](#)

Overload List

Name	Description
ApplyUpdates	Writes dataset's pending cached updates to a database.
ApplyUpdates(const UpdateRecKinds: TUpdateRecKinds)	Writes dataset's pending cached updates of specified records to a database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Writes dataset's pending cached updates to a database.

Class

[TMemDataSet](#)

Syntax

```
procedure ApplyUpdates; overload; virtual;
```

Remarks

Call the ApplyUpdates method to write a dataset's pending cached updates to a database. This method passes cached data to the database, but the changes are not committed to the database if there is an active transaction. An application must explicitly call the database component's Commit method to commit the changes to the database if the write is successful, or call the database's Rollback method to undo the changes if there is an error.

Following a successful write to the database, and following a successful call to a connection's Commit method, an application should call the CommitUpdates method to clear the cached update buffer.

Note: The preferred method for updating datasets is to call a connection component's ApplyUpdates method rather than to call each individual dataset's ApplyUpdates method. The connection component's ApplyUpdates method takes care of committing and rolling back transactions and clearing the cache when the operation is successful.

See Also

- [TMemDataSet.CachedUpdates](#)
- [TMemDataSet.CancelUpdates](#)
- [TMemDataSet.CommitUpdates](#)
- [TMemDataSet.UpdateStatus](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.3 CancelRange Method

Removes any ranges currently in effect for a dataset.

Class

[TMemDataSet](#)

Syntax

```
procedure CancelRange;
```

Remarks

Call `CancelRange` to remove a range currently applied to a dataset. Canceling a range reenables access to all records in the dataset.

See Also

- [ApplyRange](#)
- [EditRangeEnd](#)
- [EditRangeStart](#)
- [IndexFieldNames](#)
- [SetRange](#)
- [SetRangeEnd](#)
- [SetRangeStart](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.4 CancelUpdates Method

Clears all pending cached updates from cache and restores dataset in its prior state.

Class

[TMemDataSet](#)

Syntax

```
procedure CancelUpdates;
```

Remarks

Call the `CancelUpdates` method to clear all pending cached updates from cache and restore dataset in its prior state.

It restores the dataset to the state it was in when the table was opened, cached updates were last enabled, or updates were last successfully applied to the database.

When a dataset is closed, or the `CachedUpdates` property is set to `False`, `CancelUpdates` is called automatically.

See Also

- [CachedUpdates](#)
- [TMemDataSet.ApplyUpdates](#)
- [UpdateStatus](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.5 CommitUpdates Method

Clears the cached updates buffer.

Class

[TMemDataSet](#)

Syntax

```
procedure CommitUpdates;
```

Remarks

Call the CommitUpdates method to clear the cached updates buffer after both a successful call to ApplyUpdates and a database component's Commit method. Clearing the cache after applying updates ensures that the cache is empty except for records that could not be processed and were skipped by the OnUpdateRecord or OnUpdateError event handlers. An application can attempt to modify the records still in cache.

CommitUpdates also checks whether there are pending updates in dataset. And if there are, it calls ApplyUpdates.

Record modifications made after a call to CommitUpdates repopulate the cached update buffer and require a subsequent call to ApplyUpdates to move them to the database.

See Also

- [CachedUpdates](#)
- [TMemDataSet.ApplyUpdates](#)
- [UpdateStatus](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.6 DeferredPost Method

Makes permanent changes to the database server.

Class

[TMemDataSet](#)

Syntax

```
procedure DeferredPost;
```

Remarks

Call DeferredPost to make permanent changes to the database server while retaining dataset in its state whether it is dsEdit or dsInsert.

Explicit call to the Cancel method after DeferredPost has been applied does not abandon modifications to a dataset already fixed in database.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.7 EditRangeEnd Method

Enables changing the ending value for an existing range.

Class

[TMemDataSet](#)

Syntax

```
procedure EditRangeEnd;
```

Remarks

Call EditRangeEnd to change the ending value for an existing range.

To specify an end range value, call FieldByName after calling EditRangeEnd.

After assigning a new ending value, call [ApplyRange](#) to activate the modified range.

See Also

- [ApplyRange](#)
- [CancelRange](#)
- [EditRangeStart](#)
- [IndexFieldNames](#)
- [SetRange](#)
- [SetRangeEnd](#)
- [SetRangeStart](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.8 EditRangeStart Method

Enables changing the starting value for an existing range.

Class

[TMemDataSet](#)

Syntax

```
procedure EditRangeStart;
```

Remarks

Call EditRangeStart to change the starting value for an existing range.

To specify a start range value, call FieldByName after calling EditRangeStart.

After assigning a new ending value, call [ApplyRange](#) to activate the modified range.

See Also

- [ApplyRange](#)
- [CancelRange](#)
- [EditRangeEnd](#)
- [IndexFieldNames](#)
- [SetRange](#)

- [SetRangeEnd](#)
- [SetRangeStart](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.9 GetBlob Method

Retrieves TBlob object for a field or current record when only its name or the field itself is known.

Class

[TMemDataSet](#)

Overload List

Name	Description
GetBlob(Field: TField)	Retrieves TBlob object for a field or current record when the field itself is known.
GetBlob(const FieldName: string)	Retrieves TBlob object for a field or current record when its name is known.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

Retrieves TBlob object for a field or current record when its name is known.

Class

[TMemDataSet](#)

Syntax

```
function GetBlob(const FieldName: string): TBlob; overload;
```

Parameters

FieldName

Holds the name of an existing field.

Return Value

TBlob object that was retrieved.

Example

```
IBCQuery1.GetBlob('Comment').SaveToFile('Comment.txt');
```

See Also

- [TBlob](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.10 Locate Method

Searches a dataset for a specific record and positions the cursor on it.

Class

[TMemDataSet](#)

Overload List

Name	Description
Locate(const KeyFields: array of TField; const KeyValues: variant; 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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

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']), [!PartialKey]);
```

See Also

- [TMemDataSet.IndexFieldNames](#)
- [TMemDataSet.LocateEx](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.11 LocateEx Method

Excludes features that don't need to be included to the [TMemDataSet.Locate](#) method of TDataSet.

Class

[TMemDataSet](#)

Overload List

Name	Description
LocateEx(const KeyFields: array of TField; const KeyValues: variant; Options: TLocateExOptions)	Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet by the specified fields.
LocateEx(const KeyFields: string; const KeyValues: variant; Options: TLocateExOptions)	Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet by the specified field names.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.12 Prepare Method

Allocates resources and creates field components for a dataset.

Class

[TMemDataSet](#)

Syntax

```
procedure Prepare; virtual;
```

Remarks

Call the `Prepare` method to allocate resources and create field components for a dataset. To learn whether dataset is prepared or not use the `Prepared` property.

The `UnPrepare` method unprepares a query.

Note: When you change the text of a query at runtime, the query is automatically closed and unprepared.

The Prepare method is called automatically by the Open method if the dataset is not prepared.

See Also

- [Prepared](#)
- [UnPrepare](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.13 RestoreUpdates Method

Marks all records in the cache of updates as unapplied.

Class

[TMemDataSet](#)

Syntax

```
procedure RestoreUpdates;
```

Remarks

Call the RestoreUpdates method to return the cache of updates to its state before calling ApplyUpdates. RestoreUpdates marks all records in the cache of updates as unapplied. It is useful when ApplyUpdates fails.

See Also

- [CachedUpdates](#)
- [TMemDataSet.ApplyUpdates](#)
- [CancelUpdates](#)
- [UpdateStatus](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.14 RevertRecord Method

Cancels changes made to the current record when cached updates are enabled.

Class

[TMemDataSet](#)

Syntax

```
procedure RevertRecord;
```

Remarks

Call the RevertRecord method to undo changes made to the current record when cached updates are enabled.

See Also

- [CachedUpdates](#)
- [CancelUpdates](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.15 SaveToXML Method

Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Class

[TMemDataSet](#)

Overload List

Name	Description
SaveToXML(Destination: TStream)	Saves the current dataset data to a stream in the XML format compatible with ADO format.
SaveToXML(const FileName: string)	Saves the current dataset data to a file in the XML format compatible with ADO format.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

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.

See Also

- [TVirtualTable.LoadFromFile](#)
- [TVirtualTable.LoadFromStream](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.16 SetRange Method

Sets the starting and ending values of a range, and applies it.

Class

[TMemDataSet](#)

Syntax

```
procedure SetRange(const StartValues: array of System.TVarRec;  
const EndValues: array of System.TVarRec; StartExclusive: Boolean  
= False; EndExclusive: Boolean = False);
```

Parameters

StartValues

Indicates the field values that designate the first record in the range. In C++, StartValues_Size is the index of the last value in the StartValues array.

EndValues

Indicates the field values that designate the last record in the range. In C++, EndValues_Size is the index of the last value in the EndValues array.

StartExclusive

Indicates the upper and lower boundaries of the start range.

EndExclusive

Indicates the upper and lower boundaries of the end range.

Remarks

Call `SetRange` to specify a range and apply it to the dataset. The new range replaces the currently specified range, if any.

`SetRange` combines the functionality of [SetRangeStart](#), [SetRangeEnd](#), and [ApplyRange](#) in a single procedure call. `SetRange` performs the following functions:

- 1. Puts the dataset into `dsSetKey` state.
- 2. Erases any previously specified starting range values and ending range values.
- 3. Sets the start and end range values.
- 4. Applies the range to the dataset.

After a call to `SetRange`, the cursor is left on the first record in the range.

If either `StartValues` or `EndValues` has fewer elements than the number of fields in the current index, then the remaining entries are ignored when performing a search.

See Also

- [ApplyRange](#)
- [CancelRange](#)
- [EditRangeEnd](#)
- [EditRangeStart](#)
- [IndexFieldNames](#)
- [KeyExclusive](#)
- [SetRangeEnd](#)
- [SetRangeStart](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.17 SetRangeEnd Method

Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.

Class

[TMemDataSet](#)

Syntax

```
procedure SetRangeEnd;
```

Remarks

Call SetRangeEnd to put the dataset into dsSetKey state, erase any previous end range values, and set them to NULL.

Subsequent field assignments made with FieldByName specify the actual set of ending values for a range.

After assigning end-range values, call [ApplyRange](#) to activate the modified range.

See Also

- [ApplyRange](#)
- [CancelRange](#)
- [EditRangeStart](#)
- [IndexFieldNames](#)
- [SetRange](#)
- [SetRangeStart](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.18 SetRangeStart Method

Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.

Class

[TMemDataSet](#)

Syntax

```
procedure SetRangeStart;
```

Remarks

Call `SetRangeStart` to put the dataset into `dsSetKey` state, erase any previous start range values, and set them to `NULL`.

Subsequent field assignments to `FieldByName` specify the actual set of starting values for a range.

After assigning start-range values, call [ApplyRange](#) to activate the modified range.

See Also

- [ApplyRange](#)
- [CancelRange](#)
- [EditRangeStart](#)
- [IndexFieldNames](#)
- [SetRange](#)
- [SetRangeEnd](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.19 UnPrepare Method

Frees the resources allocated for a previously prepared query on the server and client sides.

Class

[TMemDataSet](#)

Syntax

```
procedure UnPrepare; virtual;
```

Remarks

Call the `UnPrepare` method to free the resources allocated for a previously prepared query on the server and client sides.

Note: When you change the text of a query at runtime, the query is automatically closed and unprepared.

See Also

- [Prepare](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.20 UpdateResult Method

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

Class

[TMemDataSet](#)

Syntax

```
function UpdateResult: TUpdateAction;
```

Return Value

a value of the TUpdateAction enumeration.

Remarks

Call the UpdateResult method to read the status of the latest call to the ApplyUpdates method while cached updates are enabled. UpdateResult reflects updates made on the records that have been edited, inserted, or deleted.

UpdateResult works on the record by record basis and is applicable to the current record only.

See Also

- [CachedUpdates](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.3.21 UpdateStatus Method

Indicates the current update status for the dataset when cached updates are enabled.

Class

[TMemDataSet](#)

Syntax

```
function UpdateStatus: TUpdateStatus; override;
```

Return Value

a value of the TUpdateStatus enumeration.

Remarks

Call the UpdateStatus method to determine the current update status for the dataset when cached updates are enabled. Update status can change frequently as records are edited, inserted, or deleted. UpdateStatus offers a convenient method for applications to assess the current status before undertaking or completing operations that depend on the update status of the dataset.

See Also

- [CachedUpdates](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.4 Events

Events of the **TMemDataSet** class.

For a complete list of the **TMemDataSet** class members, see the [TMemDataSet Members](#) topic.

Public

Name	Description
OnUpdateError	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](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.4.1 OnUpdateError Event

Occurs when an exception is generated while cached updates are applied to a database.

Class

[TMemDataSet](#)

Syntax

```
property OnUpdateError: TUpdateErrorEvent;
```

Remarks

Write the OnUpdateError event handler to respond to exceptions generated when cached updates are applied to a database.

E is a pointer to an EDatabaseError object from which application can extract an error message and the actual cause of the error condition. The OnUpdateError handler can use this information to determine how to respond to the error condition.

UpdateKind describes the type of update that generated the error.

UpdateAction indicates the action to take when the OnUpdateError handler exits. On entry into the handler, UpdateAction is always set to uaFail. If OnUpdateError can handle or correct the error, set UpdateAction to uaRetry before exiting the error handler.

The error handler can use the TField.OldValue and TField.NewValue properties to evaluate error conditions and set TField.NewValue to a new value to reapply. In this case, set UpdateAction to uaRetry before exiting.

Note: If a call to ApplyUpdates raises an exception and ApplyUpdates is not called within the context of a try...except block, an error message is displayed. If the OnUpdateError handler cannot correct the error condition and leaves UpdateAction set to uaFail, the error message is displayed twice. To prevent redisplay, set UpdateAction to uaAbort in the error handler.

See Also

- [CachedUpdates](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.1.1.4.2 OnUpdateRecord Event

Occurs when a single update component can not handle the updates.

Class

[TMemDataSet](#)

Syntax

```
property OnUpdateRecord: TUpdateRecordEvent;
```

Remarks

Write the OnUpdateRecord event handler to process updates that cannot be handled by a single update component, such as implementation of cascading updates, insertions, or deletions. This handler is also useful for applications that require additional control over parameter substitution in update components.

UpdateKind describes the type of update to perform.

UpdateAction indicates the action taken by the OnUpdateRecord handler before it exits. On entry into the handler, UpdateAction is always set to uaFail. If OnUpdateRecord is successful, it should set UpdateAction to uaApplied before exiting.

See Also

- [CachedUpdates](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.2 Variables

Variables in the **MemDS** unit.

Variables

Name	Description
DoNotRaiseExcetionOnUaFail	An exception will be raised if the value of the UpdateAction parameter is uaFail.
SendDataSetChangeEventAfterOpen	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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.2.1 DoNotRaiseExcetionOnUaFail Variable

An exception will be raised if the value of the UpdateAction parameter is uaFail.

Unit

[MemDS](#)

Syntax

```
DoNotRaiseExcetionOnUaFail: boolean = False;
```

Remarks

Starting with IBDAC 2.20.0.12, if the [OnUpdateRecord](#) event handler sets the UpdateAction parameter to uaFail, an exception is raised. The default value of UpdateAction is uaFail. So, the exception will be raised when the value of this parameter is left unchanged.

To restore the old behaviour, set DoNotRaiseExcetionOnUaFail to True.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.24.2.2 SendDataSetChangeEventAfterOpen Variable

The DataSetChange event is sent after a dataset gets open. It was necessary to fix a problem with disappeared vertical scrollbar in some types of DB-aware grids.

Unit

[MemDS](#)

Syntax

```
sendDataSetChangeEventAfterOpen: boolean = True;
```

Remarks

Starting with IBDAC 2.20.0.11, the DataSetChange event is sent after a dataset gets open. It was necessary to fix a problem with disappeared vertical scrollbar in some types of DB-aware grids. This problem appears only under Windows XP when visual styles are enabled.

To disable sending this event, change the value of this variable to False.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.25 VirtualDataSet

This unit contains implementation of the TVirtualDataSet component.

Classes

Name	Description
TCustomVirtualDataSet	A base class for representation of arbitrary data in tabular form.
TVirtualDataSet	Dataset that processes arbitrary non-tabular data.

Types

Name	Description
TOnDeleteRecordEvent	This type is used for the E:Devart.Dac.TVirtualDataSet.OnDeleteRecord event.

TOnGetFieldValueEvent	This type is used for the E:Devart.Dac.TVirtualDataSet.OnGetFieldValue event.
TOnGetRecordCountEvent	This type is used for the E:Devart.Dac.TVirtualDataSet.OnGetRecordCount event.
TOnModifyRecordEvent	This type is used for E:Devart.Dac.TVirtualDataSet.OnInsertRecord and E:Devart.Dac.TVirtualDataSet.OnModifyRecord events.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.25.1 Classes

Classes in the **VirtualDataSet** unit.

Classes

Name	Description
TCustomVirtualDataSet	A base class for representation of arbitrary data in tabular form.
TVirtualDataSet	Dataset that processes arbitrary non-tabular data.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.25.1.1 TCustomVirtualDataSet Class

A base class for representation of arbitrary data in tabular form.

For a list of all members of this type, see [TCustomVirtualDataSet](#) members.

Unit

[virtualDataSet](#)

Syntax

```
TCustomVirtualDataSet = class(TMemDataSet);
```

Inheritance Hierarchy

[TMemDataSet](#)

TCustomVirtualDataSet

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.25.1.1.1 Members

[TCustomVirtualDataSet](#) class overview.

Properties

Name	Description
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
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
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
GetBlob (inherited from TMemDataSet)	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
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.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
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
------	-------------

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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.25.1.2 TVirtualDataSet Class

Dataset that processes arbitrary non-tabular data.

For a list of all members of this type, see [TVirtualDataSet](#) members.

Unit

[virtualDataSet](#)

Syntax

```
TVirtualDataSet = class(TCustomVirtualDataSet);
```

Inheritance Hierarchy

[TMemDataSet](#)

[TCustomVirtualDataSet](#)

TVirtualDataSet

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.25.1.2.1 Members

[TVirtualDataSet](#) class overview.

Properties

Name	Description
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates

	for a dataset.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
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
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.

DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
GetBlob (inherited from TMemDataSet)	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
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.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.

SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
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
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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.25.2 Types

Types in the **VirtualDataSet** unit.

Types

Name	Description
TOnDeleteRecordEvent	This type is used for the E:Devart.Dac.TVirtualDataSet.OnDeleteRecord event.

TOnGetFieldValueEvent	This type is used for the E:Devart.Dac.TVirtualDataSet.OnGetFieldValue event.
TOnGetRecordCountEvent	This type is used for the E:Devart.Dac.TVirtualDataSet.OnGetRecordCount event.
TOnModifyRecordEvent	This type is used for E:Devart.Dac.TVirtualDataSet.OnInsertRecord and E:Devart.Dac.TVirtualDataSet.OnModifyRecord events.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.25.2.1 TOnDeleteRecordEvent Procedure Reference

This type is used for the E:Devart.Dac.TVirtualDataSet.OnDeleteRecord event.

Unit

[virtualDataSet](#)

Syntax

```
TOnDeleteRecordEvent = procedure (Sender: Tobject; RecNo: Integer) of object;
```

Parameters

Sender

An object that raised the event.

RecNo

Number of the record being deleted.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.25.2.2 TOnGetFieldValueEvent Procedure Reference

This type is used for the E:Devart.Dac.TVirtualDataSet.OnGetFieldValue event.

Unit

[VirtualDataSet](#)

Syntax

```
TOnGetFieldValueEvent = procedure (Sender: TObject; Field: TField;  
RecNo: Integer; out Value: Variant) of object;
```

Parameters

Sender

An object that raised the event.

Field

The field, which data has to be returned.

RecNo

The number of the record, which data has to be returned.

Value

Requested field value.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.25.2.3 TOnGetRecordCountEvent Procedure Reference

This type is used for the E:Devart.Dac.TVirtualDataSet.OnGetRecordCount event.

Unit

[VirtualDataSet](#)

Syntax

```
TOnGetRecordCountEvent = procedure (Sender: TObject; out Count:  
Integer) of object;
```

Parameters

Sender

An object that raised the event.

Count

The number of records that the virtual dataset will contain.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.25.2.4 TOnModifyRecordEvent Procedure Reference

This type is used for E:Devart.Dac.TVirtualDataSet.OnInsertRecord and E:Devart.Dac.TVirtualDataSet.OnModifyRecord events.

Unit

[virtualDataSet](#)

Syntax

```
TOnModifyRecordEvent = procedure (Sender: TObject; var RecNo: Integer) of object;
```

Parameters

Sender

An object that raised the event.

RecNo

Number of the record being inserted or modified.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.26 VirtualTable

This unit contains implementation of the TVirtualTable component.

Classes

Name	Description
TVirtualTable	Dataset that stores data in memory. This component is placed on the Data Access page of the Component palette.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.26.1 Classes

Classes in the **VirtualTable** unit.

Classes

Name	Description
TVirtualTable	Dataset that stores data in memory. This component is placed on the Data Access page of the Component palette.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.26.1.1 TVirtualTable Class

Dataset that stores data in memory. This component is placed on the Data Access page of the Component palette.

For a list of all members of this type, see [TVirtualTable](#) members.

Unit

[virtualTable](#)

Syntax

```
TVirtualTable = class(TMemDataSet);
```

Inheritance Hierarchy

[TMemDataSet](#)

TVirtualTable

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.26.1.1.1 Members

[TVirtualTable](#) class overview.

Properties

Name	Description
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
DefaultSortType	Used to determine the default type of local sorting for string fields.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
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
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
Assign	Copies fields and data from another TDataSet component.

CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
GetBlob (inherited from TMemDataSet)	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
LoadFromFile	Loads data from a file into a TVirtualTable component.
LoadFromStream	Copies data from a stream into a TVirtualTable component.
Locate (inherited from TMemDataSet)	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
LocateEx (inherited from TMemDataSet)	Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.
Prepare (inherited from TMemDataSet)	Allocates resources and creates field components for a dataset.
RestoreUpdates (inherited from TMemDataSet)	Marks all records in the cache of updates as unapplied.
RevertRecord (inherited from TMemDataSet)	Cancels changes made to the current record when

	cached updates are enabled.
SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
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
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.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.26.1.1.2 Properties

Properties of the **TVirtualTable** class.

For a complete list of the **TVirtualTable** class members, see the [TVirtualTable Members](#) topic.

Public

Name	Description
CachedUpdates (inherited from TMemDataSet)	Used to enable or disable the use of cached updates for a dataset.
IndexFieldNames (inherited from TMemDataSet)	Used to get or set the list of fields on which the recordset is sorted.
KeyExclusive (inherited from TMemDataSet)	Specifies the upper and lower boundaries for a range.
LocalConstraints (inherited from TMemDataSet)	Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.
LocalUpdate (inherited from TMemDataSet)	Used to prevent implicit update of rows on database server.
Prepared (inherited from TMemDataSet)	Determines whether a query is prepared for execution or not.
Ranged (inherited from TMemDataSet)	Indicates whether a range is applied to a dataset.
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.

Published

Name	Description
DefaultSortType	Used to determine the default type of local sorting for string fields.

See Also

- [TVirtualTable Class](#)
- [TVirtualTable Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.26.1.1.2.1 DefaultSortType Property

Used to determine the default type of local sorting for string fields.

Class

[TVirtualTable](#)

Syntax

```
property DefaultSortType: TSortType default stCaseSensitive;
```

Remarks

The DefaultSortType property is used when a sort type is not specified explicitly after the field name in the [TMemDataSet.IndexFieldNames](#) property of a dataset.

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.26.1.1.3 Methods

Methods of the **TVirtualTable** class.

For a complete list of the **TVirtualTable** class members, see the [TVirtualTable Members](#) topic.

Public

Name	Description
ApplyRange (inherited from TMemDataSet)	Applies a range to the dataset.
ApplyUpdates (inherited from TMemDataSet)	Overloaded. Writes dataset's pending cached updates to a database.
Assign	Copies fields and data from another TDataSet component.
CancelRange (inherited from TMemDataSet)	Removes any ranges currently in effect for a dataset.
CancelUpdates (inherited from TMemDataSet)	Clears all pending cached updates from cache and restores dataset in its prior state.
CommitUpdates (inherited from TMemDataSet)	Clears the cached updates buffer.
DeferredPost (inherited from TMemDataSet)	Makes permanent changes to the database server.
EditRangeEnd (inherited from TMemDataSet)	Enables changing the ending value for an existing range.
EditRangeStart (inherited from TMemDataSet)	Enables changing the starting value for an existing range.
GetBlob (inherited from TMemDataSet)	Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.
LoadFromFile	Loads data from a file into a TVirtualTable component.
LoadFromStream	Copies data from a stream into a TVirtualTable component.
Locate (inherited from TMemDataSet)	Overloaded. Searches a dataset for a specific record and positions the cursor on it.
LocateEx (inherited from TMemDataSet)	Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate

	method of TDataSet.
Prepare (inherited from TMemDataSet)	Allocates resources and creates field components for a dataset.
RestoreUpdates (inherited from TMemDataSet)	Marks all records in the cache of updates as unapplied.
RevertRecord (inherited from TMemDataSet)	Cancels changes made to the current record when cached updates are enabled.
SaveToXML (inherited from TMemDataSet)	Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.
SetRange (inherited from TMemDataSet)	Sets the starting and ending values of a range, and applies it.
SetRangeEnd (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.
SetRangeStart (inherited from TMemDataSet)	Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.
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.

See Also

- [TVirtualTable Class](#)
- [TVirtualTable Class Members](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.26.1.1.3.1 Assign Method

Copies fields and data from another TDataSet component.

Class

[TVirtualTable](#)

Syntax

```
procedure Assign(Source: TPersistent); override;
```

Parameters

Source

Holds the TDataSet component to copy fields and data from.

Remarks

Call the Assign method to copy fields and data from another TDataSet component.

Note: Unsupported field types are skipped (i.e. destination dataset will contain less fields than the source one). This may happen when Source is not a TVirtualTable component but some server-oriented dataset.

Example

```
Query1.SQL.Text := 'SELECT * FROM DEPT';  
Query1.Active := True;  
VirtualTable1.Assign(Query1);  
VirtualTable1.Active := True;
```

See Also

- [TVirtualTable](#)

© 1997-2024

Devart. All Rights Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.26.1.1.3.2 LoadFromFile Method

Loads data from a file into a TVirtualTable component.

Class

[TVirtualTable](#)

Syntax

```
procedure LoadFromFile(const FileName: string; LoadFields:  
boolean = True; DecodeHTMLEntities: boolean = True);
```

Parameters

FileName

Holds the name of the file to load data from.

LoadFields

Indicates whether to load fields from the file.

DecodeHTMLEntities

Indicates whether to decode HTML entities from the file.

Remarks

Call the LoadFromFile method to load data from a 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. The file format is detected automatically.

© 1997-2024

Devart. All Rights
Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)

5.26.1.1.3.3 LoadFromStream Method

Copies data from a stream into a TVirtualTable component.

Class

[TVirtualTable](#)

Syntax

```
procedure LoadFromStream(Stream: TStream; LoadFields: boolean =  
True; DecodeHTMLEntities: boolean = True);
```

Parameters*Stream*

Holds the stream from which the field's value is copied.

LoadFields

Indicates whether to load fields from the stream.

DecodeHTMLEntities

Indicates whether to decode HTML entities from the stream.

Remarks

Call the LoadFromStream method to copy data from 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. The data format is detected automatically.

© 1997-2024

Devart. All Rights

Reserved.

[Request Support](#)

[DAC Forum](#)

[Provide Feedback](#)